Combined Refraction/Diffraction Model

REF/DIF 1

Version 2.5

Documentation and User's Manual

James T. Kirby and Robert A. Dalrymple

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware, Newark, DE 19716

CACR Report No. 94-22

December, 1994
Supercedes Version 2.4 (Report No. 92-04, June, 1993)

Contents

List of Figures

1 THEORETICAL BACKGROUND

The propagation of water waves over irregular bottom bathymetry and around islands involves many processes – shoaling, refraction, energy dissipation and diffraction. Until recently, only very approximate models existed to predict the wave behavior due to these effects. This manual describes the weakly nonlinear combined refraction and diffraction model initialy developed by Kirby and Dalrymple (1983a), which incorporates all of the effects mentioned above.

The practical analysis of the refraction of water waves has generally been carried out in the past using ray tracing techniques. This technique does not include wave diffraction, and therefore it is inaccurate whenever diffraction effects are important. Often due to complexities in the bottom topography, wave tracing diagrams have many intersecting wave rays which leads to difficulties in interpretation, as the theory predicts infinite wave heights at these locations. Recently, finite difference refraction models have been developed which have the advantage of providing wave heights and directions on a model grid rather than on irregularly spaced rays (see, for example, Dalrymple (1988, 1991)).

The diffraction of water waves around simple structures such as an offshore breakwater has been obtained analytically for a constant water depth, Sommerfeld (1896). Diagrams of the wave heights in the vicinity of such a structure have been presented by the Corps of Engineers (1973, also Wiegel, 1962). For a cylindrical structure, MacCamy and Fuchs (1954) presented the constant depth solution. These solutions give not only the wave height transmitted past the structure, but also the scattered, or reflected, wave radiating away from the structure. Generalized versions of these diffraction problems, using numerical techniques and the Green's function method, have yielded very powerful procedures for wave force calculation for cases where the drag force is much smaller than the inertia force.

In order to incorporate diffractive effects, the general practice has been to suspend refraction in areas where diffraction is dominant and only permit diffraction there, using Sommerfeld's analytic solution for a flat bottom. Far from the diffraction area, refraction is resumed. This ad-hoc technique clearly is inaccurate, but does permit the inclusion of diffraction in an approximate way.

Combined refraction/diffraction models include both effects explicitly, thus permitting the modelling of waves in regions where the bathymetry is irregular and where diffraction is important. Regions where wave rays cross due to local focussing or where caustics are caused by other means are treated correctly by the models and no infinite wave heights are predicted. The models, developed in parabolic form, do not predict the waves which are scattered upwave; that is, waves which are reflected directly back the way they came are not modelled and are neglected. This means, in general, wave reflection phenomena are not reproduced correctly.

Combined refraction/diffraction models are uniquely suited for the calculation of wave heights and wave direction in areas where one or both of these effects are present. Examples include the determination of wave heights in a bay given the offshore wave heights, periods and directions, and determination of the amount of wave energy penetrating an island chain, or calculation of the sheltering and hence the disturbance of the littoral processes by an island situated near a shoreline. They are not intended to replace diffraction theories currently in use for wave force calculations.

The weakly nonlinear combined refraction and diffraction model described here, denoted **REF/DIF 1**, is based on a Stokes expansion of the water wave problem and includes the third order correction to the wave phase speed. The wave height is known to second order (Liu and Tsay (1984)). It should be noted that it is not a complete third order theory, as all the third order terms are not retained. Known ambient currents, which effect the height and direction of wave propagation, are input for the model and enable it to predict waves where currents may be strong.

The application of the theoretical model to practical situations involves the use of a parabolic approximation, which restricts the model to cases where the wave propagation direction is within $\pm 60°$ of the assumed wave direction, and the use of finite difference techniques for the wave amplitude, which results in tridiagonal matrices, which are computationally very fast to invert.

The **REF/DIF 1** model is described in detail in this manual, which also documents the application of the model to actual examples and provides explicit descriptions of the input and output.

1.1 Wave Models

1.1.1 Mild slope equation

The problem of water waves propagating over irregular bathymetry in arbitrary directions is a three-dimensional problem and involves complicated nonlinear boundary condition. Very few solutions to the three dimensional problem exist and those that do are only for flat bottoms. In one horizontal dimension, sophisticated models by Chu and Mei (1970) and Djordjevic and Redekopp (1978) predict the behavior of Stokes waves over slowly varying bathymetry. In order to simplify the problem in three dimensions, Berkhoff (1972), noted that the important properties of linear progressive water waves could be predicted by a weighted vertically integrated model. (The vertical integration reduces the problem to only the two horizontal dimensions, x and y.)

Berkhoff's equation is known as the mild slope equation. It is written in terms of the surface displacement, $\eta(x, y)$. The equation, in terms of horizontal gradient operator, is

$$\nabla_h \cdot (CC_g \nabla_h \eta) + \sigma^2 \frac{C_g}{C} \eta = 0 \tag{1}$$

Here,

$$\begin{aligned} C &= \sqrt{(g/k)\tanh kh}, \text{ the wave celerity,} \\ C_g &= C\{1 + 2kh/\sinh 2kh\}/2, \text{ the group velocity,} \end{aligned}$$

where the local water depth is $h(x, y)$ and g is the acceleration of gravity. The local wave number, $k(x, y)$, is related to the angular frequency of the waves, σ, and the water depth h by the linear dispersion relationship,

$$\sigma^2 = gk \tanh kh \tag{2}$$

The model equation (1) is an approximation; however, it is quite good even for moderately large local bottom slopes (see Booij, 1983). In both deep and shallow water, it is exact. Numerous authors have applied the mild slope model to various examples, primarily using finite element techniques. See, for example, Jonsson and Skovgaard (1979), Bettess and Zienkiewicz (1977), and Houston (1981).

For the linear mild slope equation, Radder (1979) developed a parabolic model, which had several advantages over the elliptic form presented by Berkhoff. First, the boundary condition at the downwave end of the model area is no longer necessary and, secondly, very efficient solution techniques are available for the finite difference form of the model. Radder used a splitting matrix approach, which involves separating the wave field into a forward propagating wave and a backward propagating wave, and then neglecting the backward scattered wave (which is justified in most applications as only the forward propagating wave is used for design). Radder's approximation for derivatives transverse to the wave direction results in a restriction on his parabolic model: the waves must propagate within 45° of the assumed wave direction. Booij (1981) also developed a splitting of the elliptic equation, but his procedure included more terms in the approximation to the lateral derivative and therefore his procedure enables the parabolic model to handle waves propagating within 60° of the assumed direction. Booij's procedure has been used in previous versions of the **REF/DIF 1** model (up through version 2.3).

More recently, Kirby (1986b) has developed an extension to the Booij approximation based on a Minimax principle, which further extends the range of validity of the model equations. The wave-current version of the resulting model is included here for the first time, and represents one of the chief enhancements in Version 2.4.

1.1.2 Diffraction models

In contrast to the mild slope model which is valid for varying bathymetry, researchers in the area of wave diffraction were developing models for constant bottom applications. For example, Mei and Tuck (1980) developed a simple parabolic equation for wave diffraction and applied it to the diffraction of waves by a slender island. Their equation is

$$\frac{\partial A}{\partial x} = \frac{i}{2k} \frac{\partial^2 A}{\partial y^2} \tag{3}$$

where A is a *complex amplitude* related to the water surface displacement by

$$\eta = Ae^{i\,(kx - \sigma t)} \tag{4}$$

Yue and Mei (1980), using a multiple scales approach, developed a nonlinear form of this equation, which accurately predicts the propagation of a third order Stokes wave. A striking result of their numerical experiments was the development of Mach stem reflection due to the reflection of obliquely incident waves from a breakwater. This phenomenon is uniquely a nonlinear effect and not predictable from a linear refraction theory.

The parabolic model described below combines the essential features of the two approaches described above. The variable depth features of the mild-slope equation (along with extensions to include effects of wave-current interaction) are retained, but the model is developed in parabolic form and in terms of a complex amplitude A.

1.1.3 Nonlinear combined refraction/diffraction models

Kirby (1983), using a Lagrangian approach, and Kirby and Dalrymple (1983a), with a multiple scales technique, developed the predecessor to the **REF/DIF 1** model, which bridged the gap between the nonlinear diffraction models and the linear mild slope equation. This model can be written in several forms depending on the application. The hyperbolic form, for time dependent applications, and the elliptic form, for steady state problems, require the use of boundary conditions on all sides of the model domain. This is a difficult requirement, as the reflected wave at a boundary is not generally known a priori. These models, however, have the advantage that there is no restriction on the wave direction.

A detailed comparison of results of the weakly-nonlinear model of Kirby and Dalrymple(1983a) to laboratory data was shown by Kirby and Dalrymple (1984). The laboratory test, conducted at the Delft Hydraulics Laboratory by Berkhoff, Booij and Radder (1982), consisted of determining the wave amplitude over a shoal on a sloping bottom. While results predicted by ray tracing techniques were shown by Berkhoff, Booij and Radder to be very poor, the agreement between the weakly-nonlinear model and the laboratory data was excellent. Comparisons between linear and nonlinear parabolic models clearly showed the importance of the nonlinear dispersion terms in the governing equations.

1.1.4 Wave-current interaction models

Booij (1981), using a Lagrangian approach, developed a version of the mild slope equation including the influence of current. This model is a weak current model in that the currents are assumed to be small and any products of currents are neglected as small. Kirby (1984) presented the corrected form of this mild slope model. A nonlinear correction and the ability to handle strong currents were added by Kirby and Dalrymple (1983b) and results for waves interacting with a current jet were obtained. Their equation is

$$\begin{aligned}(C_g+U)A_x+VA_y \quad &+ \quad i(\bar{k}-k)(C_g+U)A+\frac{\sigma}{2}\left\{\left(\frac{C_g+U}{\sigma}\right)_x+\left(\frac{V}{\sigma}\right)_y\right\}A \\ &- \quad \frac{i}{2\sigma}\left((p-V^2)A_y\right)_y-\sigma\frac{k^2}{2}D|A|^2A=0 \end{aligned} \tag{5}$$

where $p = CC_g$ and $\bar{k}$ = reference wave number, taken as the average wave number along the y axis, and U is the mean current velocity in the x coordinate direction and V is in the y direction. The nonlinear term includes D, which is

$$D=\frac{(\cosh 4kh+8-2\tanh^2 kh)}{8\sinh^4 kh}$$

Kirby (1986a) rederived the above equation for a wide angle parabolic approximation, which allows the study of waves with larger angles of wave incidence with respect to the x axis. This more accurate equation was used as the basis for earlier versions of **REF/DIF 1**. The equation has been extended to include the more accurate minimax approximation (Kirby, 1986b) for the present version of **REF/DIF 1**. The revised governing equation is given by

$$\begin{aligned} &(C_g+U)A_x-2\Delta_1 VA_y+i(\bar{k}-a_0k)(C_g+U)A+\left\{\frac{\sigma}{2}\left(\frac{C_g+U}{\sigma}\right)_x-\Delta_1\sigma\left(\frac{V}{\sigma}\right)_y\right\}A \\ + \quad & i\Delta'\left[(p-V^2)\left(\frac{A}{\sigma}\right)_y\right]_y-i\Delta_1\left\{\left[UV\left(\frac{A}{\sigma}\right)_y\right]_x+\left[UV\left(\frac{A}{\sigma}\right)_x\right]_y\right\} \\ + \quad & \frac{i\sigma k^2}{2}D|A|^2A+\frac{w}{2}A+\frac{-b_1}{k}\left\{\left[(p-V^2)\left(\frac{A}{\sigma}\right)_y\right]_{yx}+2i\left(\sigma V\left(\frac{A}{\sigma}\right)_y\right)_x\right\} \\ + \quad & b_1\beta\left\{2i\omega U\left(\frac{A}{\sigma}\right)_x+2i\sigma V\left(\frac{A}{\sigma}\right)_y-2UV\left(\frac{A}{\sigma}\right)_{xy}+\left[(p-V^2)\left(\frac{A}{\sigma}\right)_y\right]_y\right\} \\ - \quad & \frac{i}{k}b_1\left\{(\omega V)_y+3(\omega U)_x\right\}\left(\frac{A}{\sigma}\right)_x-\Delta_2\left\{\omega U\left(\frac{A}{\sigma}\right)_x+\frac{1}{2}\omega U_x\left(\frac{A}{\sigma}\right)\right\} \\ + \quad & ik\omega U(a_0-1)\left(\frac{A}{\sigma}\right)=0 \end{aligned} \tag{6}$$

where

$$\beta = \frac{k_x}{k^2} + \frac{(k(p-U^2))_x}{2k^2(p-U^2)} \tag{7}$$

$$\Delta_1 = a_1 - b_1 \tag{8}$$

$$\Delta_2 = 1 + 2a_1 - 2b_1 \tag{9}$$

$$\Delta' = a_1 - b_1\frac{\bar{k}}{k} \tag{10}$$

and w is a dissipation factor discussed in the next section. The coefficients a_0, a_1 and b_1 depend on the *aperture width* chosen to specify the minimax approximation; see Kirby (1986). The combination

$$\begin{aligned} a_0 &= 1 \\ a_1 &= -0.5 \\ b_1 &= 0 \end{aligned} \tag{11}$$

recovers Radder's approximation, while the choices

$$\begin{aligned} a_0 &= 1 \\ a_1 &= -0.75 \\ b_1 &= -0.25 \end{aligned} \tag{12}$$

recover the approximation of Booij (1981). The values of a_0, a_1 and b_1 used for the Minimax approximation depends on the range of angles to be considered; Kirby (1986b) found that the values for a maximum angular range of 70° gave reasonable results over the range of angles typically used. The corresponding coefficient values for this choice are

$$\begin{aligned} a_0 &= 0.994733 \\ a_1 &= -0.890065 \\ b_1 &= -0.451641 \end{aligned} \tag{13}$$

Equation (6) is the model equation used in **REF/DIF 1**. At present, the model still utilizes the Padé approximation form based on the coefficients in (12); testing is underway to extend the model to the minimax model with coefficients (13).

In the previous two equations, the dispersion relationship relating the angular frequency of the wave, the depth and the wave number is changed to reflect the Doppler shift due to currents. The new form of eq. (2) is

$$(\omega - kU)^2 = gk\tanh kh \tag{14}$$

where the absolute frequency, ω, is related to the intrinsic frequency, σ, by

$$\omega = \sigma + kU \tag{15}$$

where the assumption that the wave is primarily travelling in the x direction has been used.

1.2 Assumptions

The **REF/DIF 1** model, in parabolic form, has a number of assumptions inherent in it and it is necessary to discuss these directly. These assumptions are:

1. Mild bottom slope. The mathematical derivation of the model equations assumes that the variations in the bottom occur over distances which are long in comparison to a wave length. For the linear model, Booij (1983) performed a comparison between an accurate numerical model and the mild slope model for waves shoaling on a beach. He found that for bottom slopes up to 1:3 the mild slope model was accurate and for steeper slopes it still predicted the trends of wave height changes and reflection coefficients correctly.

2. Weak nonlinearity. Strictly, the model is based on a Stokes perturbation expansion and is therefore restricted to applications where Stokes waves are valid. A measure of the nonlinearity is the Ursell parameter which is given as

$$U = HL^2/h^3 \tag{16}$$

   When this parameter exceeds 40, then the Stokes solution is no longer valid. In order to provide a model which is valid in much shallower water, a heuristic dispersion relationship developed by Hedges (1976) is provided as an option in the model. This relationship between the frequency and the water depth is

$$\sigma^2 = gk \tanh\left(kh(1 + |A|/h)\right) \tag{17}$$

   In shallow water, this equation matches that of a solitary wave, while in deep water it asymptotically approaches the linear wave result, neglecting real amplitude dispersive effects. For this reason, a model with a dispersion relationship which is a smooth patch between the Hedges form (valid in shallow water) and the Stokes relationship (valid in deep water) is used. This hybrid model is described in Kirby and Dalrymple (1986b). There are, as a result of the different dispersion relationships possible, three options in **REF/DIF 1**: (1), a linear model, (2), a Stokes-to-Hedges nonlinear model, and (3), a Stokes model. Of these options, the second will cover a broader range of water depths and wave heights than the others.

3. The wave direction is confined to a sector $\pm 70°$ to the principal assumed wave direction, due to the use of the minimax wide angle parabolic approximation of Kirby (1986b).

1.3 Energy Dissipation

1.3.1 General form

Energy dissipation in the model occurs in a number of ways depending on the situation being modelled. An energy loss term, due to Booij (1981) and expanded by Dalrymple *et al.* (1984a), permits the model to treat bottom frictional losses due to rough, porous or viscous bottoms, surface films, and wave breaking. The linear form of the mild slope equation with dissipation is

$$\frac{\partial A}{\partial x} = \frac{i}{k}\frac{\partial^2 A}{\partial y^2} + wA \tag{18}$$

where the dissipation factor, w, is given by a number of different forms depending on the nature of the energy dissipation. The factor w is the energy dissipation divided by the energy and has the units of time^{-1}.

1.3.2 Laminar surface and bottom boundary layers

At the water surface and at the bottom, boundary layers occur due to the action of viscosity. For a contaminated surface, resulting from surface films (natural or otherwise), a significant amount of damping occurs, which is dependent on the value of the fluid viscosity. From Phillips (1966), the surface film damping is

$$w = \frac{\sigma k \sqrt{(\nu/2\sigma)}(1-i)}{\tanh kh} \tag{19}$$

where ν is the kinematic viscosity. The term under the square root sign is related to the thickness of the boundary layer, which is generally small. At the bottom, the boundary layer damping is

$$w = \frac{2\sigma k \sqrt{(\nu/2\sigma)}(1-i)}{\sinh 2kh} \tag{20}$$

By setting the input switch, *iff(3)=1* in the **REF/DIF 1** model, surface and bottom damping are computed at all locations in the model.

1.3.3 Turbulent bottom boundary layer

In the field, the likely wave conditions are such that the bottom boundary layer is turbulent. In this case, an alternative specification of the energy dissipation must be provided. Utilizing a Darcy-Weisbach friction factor, f, the dissipation term can be shown to be

$$w = \frac{2\sigma k f|A|(1-i)}{3\pi \sinh 2kh \sinh kh} \tag{21}$$

See Dean and Dalrymple (1984). In order to implement this damping term, the value of $f = 0.01$ was assumed. In **REF/DIF 1**, if the input data switch *iff*(1)=1, then turbulent damping is computed at all locations.

1.3.4 Porous sand

Most sea bottoms are porous and the waves induce a flow into the bed. This results in wave damping due to the Darcy flow in the sand. For beds characterized by a given coefficient of permeability, Cp, the damping can be shown to be

$$w = \frac{gkC_p(1-i)}{\cosh^2 kh} \tag{22}$$

The coefficient of permeability, C_p, has the units of (m^2) and is of order 4.5 x 10^{-11} m^2. Liu and Dalrymple (1984) show, for very permeable sands, that the damping is inversely related to C_p and a different w term must be used. However, this case is not likely to occur in nature. Porous bottom damping is computed in **REF/DIF 1** when *iff*(2) = 1.

1.3.5 Wave breaking

For wave breaking, the model is more complicated. Dally *et al.* (1985) showed that the rate of loss of wave energy flux is dependent on the excess of energy flux over a stable value. This model has been tested for laboratory data for a number of different bottom slopes and predicts the wave height in the surf zone extremely well. Kirby and Dalrymple (1986a) show that the dissipation due to wave breaking is given as

$$w = \frac{KC_g\left(1-(\gamma h/H)^2\right)}{h} \tag{23}$$

where K and γ are empirical constants, determined by Dally *et al.* to be equal to 0.017 and 0.4 respectively. Here, the wave height, $H = 2|A|$. By using this dissipation model and a breaking index relation ($H > 0.78h$) to determine the onset of breaking, the **REF/DIF 1** model is able to represent waves both outside and inside of a surf zone. The wave breaking algorithm is always active in the model.

Large surface piercing islands and causeways which would have surf zones are handled by the 'thin film' technique of Dalrymple, Kirby and Mann (1984b) and Kirby and Dalrymple (1986a). This procedure permits the easy computation of wave heights around arbitrarily shaped islands by replacing islands with shoals of extremely shallow depth (1 cm). The wave breaking routine reduces the wave heights over the shoal to less than one half centimeter, which results in a wave which carries negligible energy and therefore no longer affects any physical processes. Thus, the **REF/DIF 1** model does not distinguish between islands and deeper water computationally. However, the model output clearly shows the influence of the islands, as will be shown in section 3. Examples of wave breaking and the combined refraction/diffraction model appear in Kirby and Dalrymple (1986a) and Dalrymple *et al.* (1984b).

1.4 Wave Climate

1.4.1 Monochromatic waves

While the **REF/DIF 1** model is typically used with monochromatic wave trains propagating in one given direction, there is no intrinsic restriction to this case. As an example, for a given frequency, the wave direction is determined by the initial wave height distribution provided by the user on the offshore grid row, corresponding to $x = 0$. As this row is parallel to the y axis, the wave is generally prescribed as

$$A(0, y) = A_0 e^{i\ell y} \tag{24}$$

where A_0 is the given wave amplitude and ℓ is the wave number in the y direction. The ℓ is related to the wave number k by the relationship, $\ell = k \sin\theta$, where θ is the angle made by the wave to the x axis. This case is obtained by using the data switches, *iwave* and *nwavs* set to one.

1.4.2 Discrete directional waves

For several waves with different directions at a given frequency, the following relationship could be used for the initial wave condition:

$$A(0, y) = \sum_{n=1}^{nwavs} A_n e^{i\ell n y} \tag{25}$$

The **REF/DIF 1** model is equipped to calculate the wave field produced by this boundary condition for a large number of user-supplied A_ns and θ_ns (up to 50). This mode is accessed by *iwave=1* and *nwavs* set to the number of discrete waves to be used.

1.4.3 Directional spectrum

Often, a $\cos^{2n}\theta$ directional spread is used with a given frequency component. This can be done with **REF/DIF 1** by specifying *iwave* equal to 2 and *nwavs* to the value of n. The total energy at frequency, σ, is

$$E(\sigma) = \int_0^{2\pi} E_n \ \cos^{2n}(\frac{\theta - \theta_o}{2}) \tag{26}$$

In order to avoid the problem of waves propagating at large angles to the propagation direction, θ_0, the directional distribution of energy is automatically truncated to include only those directions which contain more than 10% of the total energy.

To prescribe the initial conditions for the model, the directional distribution is discretized into 31 components, each with an amplitude characteristic of the waves in that particular directional band. These discrete waves are then assigned random phases and summed as in Eq. (21).

Note that this directional spectrum is for a given frequency, and not for a continuous distribution of frequencies as in a true directional spectrum. At the present time, the **REF/DIF 1** model can only calculate waves at a single frequency per calculation. The model will compute numerous frequencies per computer run (set *nfreqs* greater than one); however, they are not superimposable, as the wave-wave interactions between different frequencies are not included. Using the linear mode (*ntype*=0) and superimposing the results will provide a linear approximation to a directional spectrum.

The problem of computing the shoreward evolution of a directional spectrum of refracting, diffracting and breaking waves has recently been addressed in a new model called **REF/DIF S**. This model is essentially an enhancement to **REF/DIF 1** which allows for the simultaneous computation of many wave components on a large vectorizing computer. Since the amount of computer power required to run **REF/DIF S** in full spectral mode is enormous, the single component model **REF/DIF 1** will continue to be supported and developed as well.

1.5 Numerical Development

1.5.1 Crank-Nicolson Technique

The parabolic model is conveniently solved in finite difference form. In order to accomplish this, the study area bathymetry must be input as a grid with the (x, y) directions, divided into rectangles of Δx and Δy sizes. The complex amplitude $A(x, y)$ will then be sought at each grid and therefore we can keep track of A by denoting its location, not by (x, y), but by (i, j) where $x = (i-1)\Delta x$ and $y = (j-1)\Delta y$. Now we have to determine the values of $A(i,j)$ which satisfy Eq.(6) for all i between 1 and m and for all j between 1 and n. The procedure involves expressing all the derivatives in the (x,y) directions in terms of the complex amplitude at various grid points. For example, the forward difference representation of

$$\frac{\partial A}{\partial x} = \frac{A_{i+1,j} - A_{i,j}}{\Delta x} \text{ at location } i,j \tag{27}$$

If a forward difference is used for the x direction and a central difference representation centered at i is used for the second derivatives in the lateral direction for all the derivatives in Eq. (6), then an explicit finite difference equation results for $A_{i+1,j}$. This equation can be solved directly for all the $A_{i+1,j}, j = 1,2,3...n$, for a given i, provided appropriate lateral boundary conditions are prescribed. This explicit representation is not as accurate as an implicit scheme and therefore an implicit Crank-Nicolson procedure is used for the amplitude calculations. For a given i row, the Crank-Nicolson scheme can be written

$$aA_{i+1,k+1} + bA_{i+1,j} + cA_{i+1,j-1} = dA_{i,j+1} + eA_{i,j} + fAi,j-1 \tag{28}$$

where the coefficients a,b,c,d,e,f involve variable, complex and nonlinear terms. The amplitudes on the left hand side of this equation are unknown, while the terms on the right hand side are known, from either the previous calculation or from the initial boundary condition on $j = 1$ and n. This equation is solved for all the $A_{i+1,j}, j = 2n-1$ and i fixed, at once by a tridiagonal matrix solution procedure (Carnahan, Luther and Wilkes (1969)), adapted for complex-valued coefficients. Due to the nonlinearity of the finite difference equation, nonlinear terms are approximated on a first pass by using the $A_{i,j}$ values. Once the $A_{i+1,j}$ terms are computed, the equation is solved again for $A_{i+1,j}$ using now the just-calculated values in the nonlinear terms. This two-pass iterative method insures that the nonlinearities in the model are treated accurately (Kirby and Dalrymple, 1983a). The solution proceeds by moving one grid row in the x direction (incrementing i by one) and, using the two-pass implicit-implicit technique, determining the complex amplitude $A_{i+1,j}$ for all the values of j on this row. Marching in the wave direction these calculations are repeated until all the $A_{i,j}$ are known for all i and j. While it appears that the Crank-Nicolson procedure could be time consuming, given that there is a matrix inversion for each grid row, the coefficient matrix size is only 3 by n and the matrix inversion procedure is, in fact, very fast. The procedure is economical on storage as only the values for the rows i and $i+1$ are necessary at each calculation.

1.5.2 Initial and Lateral Boundary Conditions

The initial condition is vital for the parabolic model. The furthest seaward grid row, corresponding to $i = 1$, is taken as constant depth and the incident wave(s) is prescribed here. This wave is then propagated over the bathymetry by the model. The various initial conditions were discussed in the section, Wave Climate.

As in the solution of any differential equation in a domain, the lateral boundary conditions are important. There are several ways to treat the boundaries; however, none of the presently existing boundary conditions result in the total transmission of scattered waves. Therefore, for the **REF/DIF 1** model, a totally reflecting condition is generally used for each side ($j = 1$ and n). This requires that the specification of the model grid be done with care, as the reflection of the incident wave from the lateral boundaries can propagate into the region of interest rapidly and result in erroneous results.

In general, the width of the model should be such that no reflection occurs until far downwave of the region of interest. As a precaution, a graphical representation of the computed wave field should be examined to determine where the reflection from the boundaries is important. By using the switch, *ibc*, partially transmitting boundaries can be used (Kirby, 1986c). In general, this boundary condition will result in less reflection in the model domain; however, since some reflection will occur, it is recommended that runs be carried out with the reflecting boundary conditions in order to assess the regions potentially affected by reflection from the model boundaries.

1.5.3 Subgrids

In order to reduce the amount of data input and yet provide the user the ability to prescribe the fine scale bathymetry in areas of interest, **REF/DIF 1** utilizes a coarse scale user-specified reference grid and a fine scale subgrid, which can have many times the resolution of the reference grid. The principal purpose of the subgrid is to provide enough computational points to the numerical model to preserve accuracy. The user specifies the number of subgrid divisions in the y direction with the parameter *nd*. If *nd*=1, then the subgrid spacing in the y direction is the same as the reference grid. If *nd*=2, then the model uses twice as many computational points in the y direction as there are in the reference grid. In the propagation direction, x, the model will automatically determine the subgrid spacing if *ispace* has been set to unity. Otherwise, the user provides the subgrid spacing using the input, *mr*, which permits variable spacing in the x direction. For subgrids, the input flag, *isp*, must be set to one and an array, *isd*, must be specified.

1.5.4 Damping

When the large angle parabolic approximation is used as a basis for the computation of wave fields around islands, the presence of wave breaking, and resulting sharp lateral variations in wave height, leads to the generation of high-wavenumber spectral components in the computed complex amplitude A in the lateral (y) direction. Kirby (1986a) has shown that these components have propagation velocities which can become large in an unbounded fashion; as a result, they can propagate across the grid, filling the computational domain with high-wavenumber noise.

Previous versions of **REF/DIF 1** have attempted to control the high frequency noise generated by the breking process using smoothing filters, which are applied to the computed complex amplitude in between each computational row. In the present version of **REF/DIF 1**, this procedure is being replaced by a new algorithm which has recently been developed by Kirby (1993). Reference should be made to that document for a description of the algorithm. The damping is built into the computational algorithm and is turned on automatically if breaking has started anywhere in the computational domain.

2 USER'S MANUAL

This chapter provides the operating manual for the program **REF/DIF 1**. Chapter 3 provides sample documentation and calculations for example problems. A separate Fortran program *datgenv25.f* is provided which generates the input data files for these as well as a number of additional examples. In addition, the programs *indat-create.f* (which assists the user in constructing the *indat.dat* file(and *indat-convert.f* (which converts old free-format *indat.dat* files to new namelist format *indat.dat* files) are provided.

2.1 REF/DIF 1 Revision History.

2.1.1 Changes Appearing in Version 2.0

Several changes have been made to the program released as Version 1.0 of **REF/DIF 1**. These changes are outlined here for convenience.

Directional spectra application

The routine used to specify a directional spectrum as the initial condition for the wave calculations has been extensively revised and tested. The present model is based on a Mitsuyasu-type spreading factor and apportions wave directions and energy density in randomly-sized directional bins. In addition, the original random number generator supplied with Version 1.0 was found to not be sufficiently random, and a new version is supplied with Version 2.0.

ibc **- Open Boundary Condition Parameter**

Version 2.0 contains an option to use open lateral boundary conditions, which are designed to be reasonably transparent both to entering and exiting waves, if the topography near the boundary is reasonably uniform. The theory for these conditions are contained in the reference Kirby(1986c). It is recommended that initial runs for a particular site be done with the default reflective conditions in order to see the magnitude of the boundary effects and then to use the open conditions for final runs. The open boundary condition is invoked by choosing a value of *ibc*=1 in the input data.

icur - **No Currents Parameter**

A new parameter *icur* is included in the input data, and determines whether or not the program is to read in current values from the input data files. This change provides the option of not having to include zero current values on input whenno currents are being considered.

2.1.2 Changes Appearing in Version 2.1

Version 2.1 represents a minor modification. The change consists of a revision of the input file format structures for the file *indat.dat*. The formats for this file have been replaced by free format **read** statements, so that the user can enter data separated by comments without regard to the column structure. Note that data entered in the previously defined formats will still be read properly, so existing *indat.dat* files will still work properly.

2.1.3 Changes Appearing in Version 2.2

Version 2.2 includes a revised version of a dissipation filter which is used to damp out noise after the onset of breaking in the numerical computations. This filter has been found to be much more suitable in applications to field situations than was the original filter.

In addition, Version 2.2 also provides the capability to input the first row of complex amplitude A from a new external data file *wave.dat*. The procedures for specifying *wave.dat* are described in section 2.4 of the manual.

2.1.4 Changes Appearing in Version 2.3

Version 2.3 provides a provision to save the last subdivided row of complex valued amplitudes computed on the last model grid row in file *owave.dat*. This option is potentially useful if the user wants to perform a run in several segments. *owave.dat* could then be used to store the intermediate calculation required to initialize a subsequent model run. The provision for using this option is described in section 2.5. Use of this option does not affect any internal model calculations.

2.1.5 Changes Appearing in Version 2.4

Version 2.4 incorporates two revisions to the basic model scheme. The first revision is an extension to the model equations to handle the Minimax approximation of Kirby (1986b) as well as the Padé large angle approximation (Booij, 1981; Kirby, 1986a). This algorithm is not used yet in the released version of the program.

The second revision is the replacement of the add-on noise filtering algorithm with an algorithm that functions as an imbedded part of the model equation and finite-difference approximation, as described by Kirby (1993). Additionally, several inconsistencies appearing in input error checking have been corrected. A more robust algorithm for computing wave angles (due to Medina (1991)) has been added.

2.1.6 Changes Appearing in Version 2.5

A substantial number of changes appear in Version 2.5. These changes represent a combination of data format changes and enhanced post-processing, with the addition of several new computed output variables. Since the use of Matlab is becoming more widespread, we have included a *.m* script which we are presently using to present computed results.

Many of the changes in version 2.5 were made in order to include the model in the Littoral Remote Sensing System (LRSS) developed for the U. S. Navy. In cases where the changes in data formats were consistent with the vast majority of available *Fortran 77* compilers (as in the use of *namelist* formats), the changes were made directly in **REF/DIF 1**. In cases where the standard required the use of a non-typical format (as in the use of the machine-independent binary *HDF* formats for large arrays), then the data transfer to and from LRSS is handled using a pre- and post-processing layer. The intent is that the normal user of **REF/DIF 1** should not need access to any tools beyond the usual *Fortran* compiler. We caution that *namelist* is not a standard *Fortran 77* feature; however, we have not found any compilers yet which do not provide this feature.

Revision to *indat.dat* file structure.

The most readily apparent change to the long-term user of **REF/DIF 1** is the change to the use of *namelist* to structure the *indat.dat* data file. The structure of the file and the meaning of each input variable are described in section 2.7, and the file for each example is given in Chapter 3.

The program *datgen.f* supplied with older versions of the program has been updated and renamed to *datgenv25.f*. The new version produces the *namelist* formatted *indat.dat* file. Since long term users are likely to have their own versions of old *indat.dat* files, we have also provided a new program *indat-convertv25.f* which converts old *indat.dat* files to new *indat.new* files (which should then be renamed). A quick test of the integrity of this new data file convention could be made by using the old *datgen* to generate an *indat.dat*, converting it to *indat.new*, and then comparing it to the file *indat.dat* generated by the new *datgenv25*. The two resulting files should be identical.

Use of *param.h* file to dimension arrays.

Changing dimensions in **REF/DIF 1** in the past has involved a careful search through a number of subroutines to get all *parameter* statements revised properly. This has not been a pleasant process. In addition, the specification of several array variables in *namelist* makes it necessary that the dimension of the array in the program generating the data be the same as the dimension in the program reading the data.

For this reason, we have isolated the *parameter* statement in a file *param.h*, which is then used to dimension all of the programs. This file may be edited in isolation, after which all programs which are to be used (pre- and post-processing as well as **REF/DIF 1**) should be recompiled. (For UNIX users, this updating is automated by the included *Makefile*).

Stored output data files.

Prior to version 2.5, output was directed to two files. *outat.dat* was used primarily to store the complex amplitude data, which could later be used to construct either a wave height field or an image of the instantaneous surface. Data was stored at the reference grid spacing. In instances where a large amount of internal subdividing was being done, this procedure was inadequate for the construction of a picture of the surface, since the surface undulations are not resolved at the reference grid spacing.

The remainder of output data in older versions was directed either to the screen or to a file *rundat.dat*. This included header information and error log as well as x location, reference phase, height and wave angle at each reference grid point. It has been difficult to use this information conveniently, since the appearance of a warning or error message in the output could disturb the file format.

As a result, the output from **REF/DIF 1** has been almost completely restructured in Version 2.5. Values of wave height, wave angle, water depth and (in the near future) radiation stress components are stored in separate files at the reference grid resolution. The complex amplitude data needed to construct a surface image is stored at the computational resolution. The program *surface.f* (or, in the LRSS version, *rf2hdf.f*) interpolates this data onto a regularly spaced rectangular grid and stores the surface image. These files are described in section 2.9. Finally, the user may store an estimate of the magnitude of the bottom velocity in a file *bottomu.dat*.

2.2 Overview of Operating Manual

This section provides a description of the program structure (section 2.3), followed by some notes on problems which are likely to be encountered during the installation and use of the program on different computer systems (section 2.4). Section 2.5 presents the two levels of grid information used by the program. Section 2.6 describes the option of reading in the first row of complex amplitude values A from an additional external data file *wave.dat* (This option is not available to users of **REF/DIF 1** in the LRSS system). The input data file structure is then discussed. The program reads data in two essentially separate groups. The first group of data establishes the size of the model grid and gives the wave conditions to be studied; this group is discussed in section 2.7. The second group of data gives the reference grid data values and defines any user-specified subgrids; this is discussed in section 2.8. The structure of the program output is discussed in section 2.9.

A listing of the program is included in Appendix A. Appendix B provides listings of the pre- and postprocessing programs provided with all versions of REF/DIF 1. Appendix C provides listings of additional pre-and postprocessing programs which are specific to LRSS. Finally, Appendix D provides listings for some sample plotting programs.

2.3 Program Outline and Flow Chart

The model **REF/DIF 1** is organized in one main program *refdif1* and fourteen subroutines. The program does not contain calls to any external package, and should be free standing on any system. The *LRSS* version of the program requires access to several linking libraries, including *libdf.a*, the *HDF* library provided by *NCSA*, and the libraries *libezhdf.a* and *liblrss.a*, provided by *SAIC* for the *LRSS* system. The program should be a legal code for any compiler which accepts the full FORTRAN 77 standard language. A possible exception would be the appearance of *namelist* statements, which are not part of the FORTRAN 77 standard but which are provided by all compilers we have checked.

REF/DIF 1 is structured in two levels; a main level, which reads in and checks input data and then starts the operation of the wave model level, and the model level itself, which performs the actual finite difference calculations. The flow charts for the two levels are given in Figures 1 and 2. A short description of each routine in the model follows.

1. *refdif1*: Main program controls the calls to *inref* and *inwave* to read in data, and to *model*, which performs the actual calculations. No calculations are performed by this routine.

2. *inref* : Called by *refdif1*. *inref* reads in data controlling reference grid dimensions and the grid interpolation scheme from logical device number $iun(5)$, and reads in the

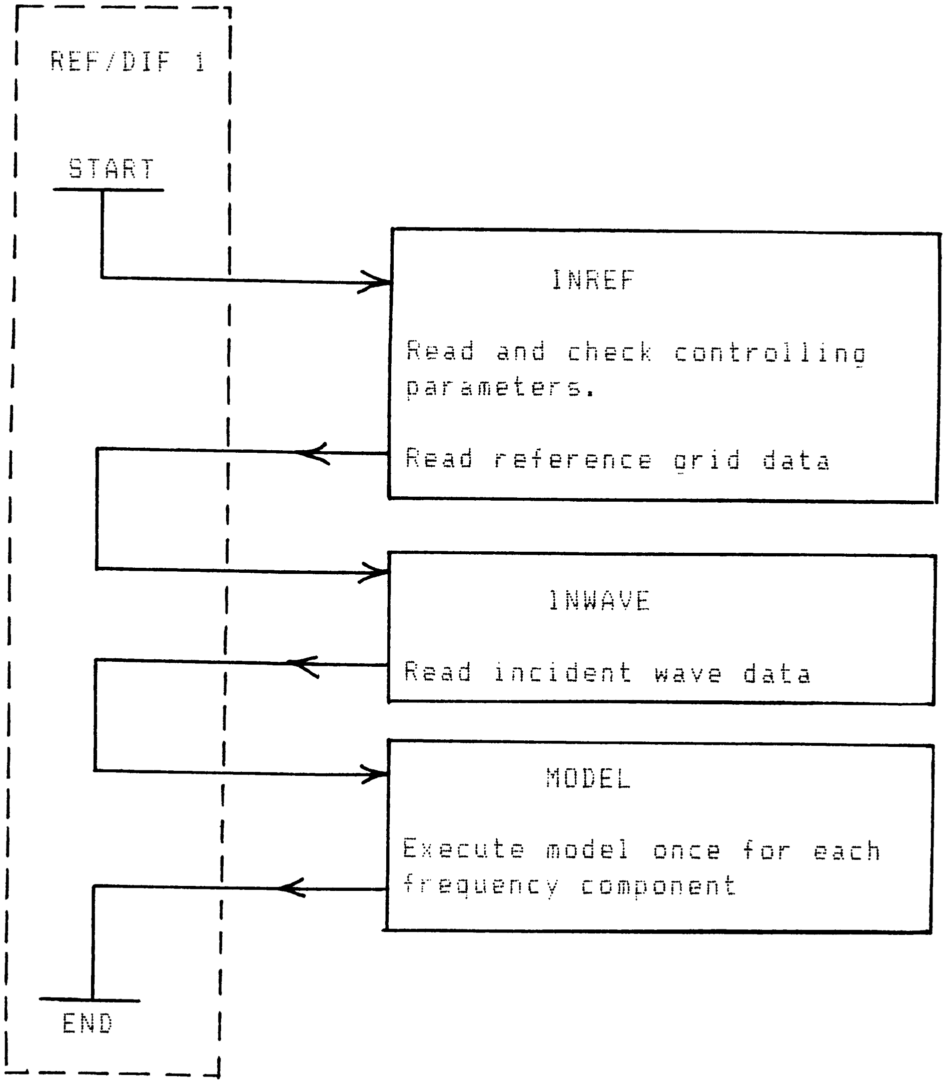

Figure 1: REF/DIF 1: *refdif1* program level

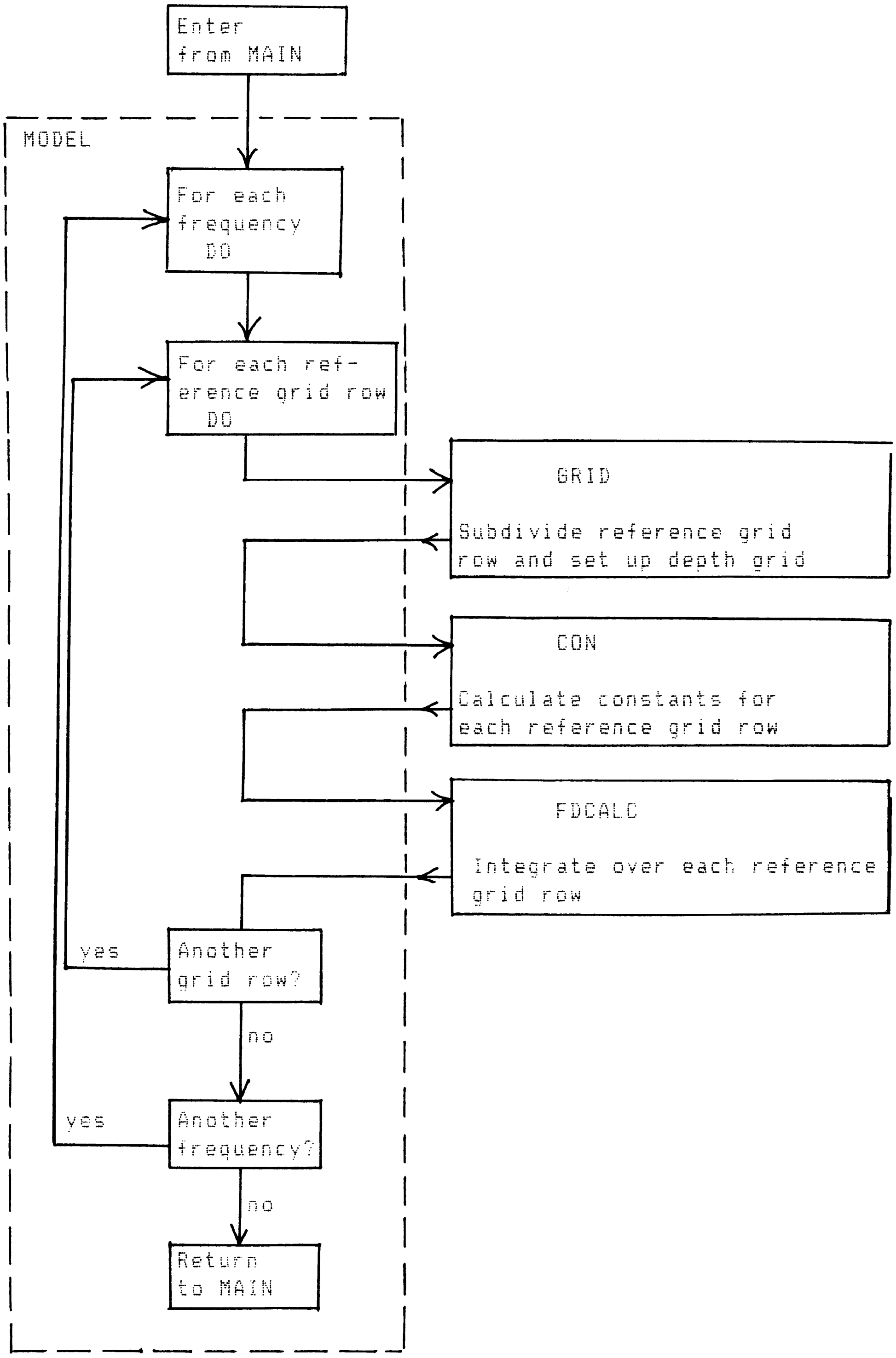

Figure 2: REF/DIF 1: *model* subroutine level

reference grid values of depth *dr*, x-direction velocity *ur* and y-direction velocity *vr* from logical device number *iun*(1). Some data checking is performed. If data is read in in English units, *inref* converts it to MKS units using the *dconv* factor. Output files are initialized. A description of the data files may be found in sections 2.6-2.8 of this manual. At the end of the subroutine, control is returned to *refdif1*.

3. *inwave* : Called by *refdif1*. *inwave* reads in data specifying the initial wave field along the first row of reference grid points. Data is read from logical device number *iun*(5). Conversion to MKS units is performed for data read in in English units. Control is returned to *refdif1*.

4. *model* : Called from *refdif1*. *model* controls execution of the computational part of the program. For each frequency component specified in the input, *model* performs the following series of operations:

    (a) Initialize program by calculating the incident wave field on the first grid row.

    (b) Then, for each grid block in the reference grid:

    - Call *grid* to perform the grid interpolation specified in the input data.
    - Call *con* to calculate constants on the interpolated grid.
    - Call *fdcalc* to perform the numerical integration of the parabolic equation over the interpolated subgrid.

    The model execution is then complete. Control is returned to *refdif1*.

5. *grid* : Called by *model*. *grid* performs the required interpolation over a single grid block of the reference grid as specified in the input data.The interpolation is performed as described in section 2.3. *grid* checks to see whether a user-specified subgrid feature should be read in, and reads it in from logical unit number *iun*(2) if needed. The interpolated depth grid is then corrected for tidal offset, and checked for surface-piercing features. These features are modified using the "thin-film" approach; see Kirby and Dalrymple (1986a). Control is returned to *model*.

6. *con* : Called by *model*. *con* calculates various constants for the reference grid created by *grid*. Control is returned to *model*.

7. *fdcalc* : Called from *model*. *fdcalc* performs the integration of the governing parabolic equation over the grid defined in *grid*. The coefficients of the finite difference form of the parabolic equation are developed according to the Crank-Nicolson method. A complete description of the equations and the treatment of nonlinearities may be

found in Kirby (1986a) and Kirby and Dalrymple (1986b). The sequence of steps in *fdcalc* is as follows:

(a) An implicit step is performed to update complex amplitude A along an entire grid row.

(b) The model checks for the start or stop of breaking on the updated row.

(c) If the status of breaking changes, the model recomputes the breaking wave dissipation coefficient.

(d) Then, if nonlinearity is being used or breaking status at any point along the row has changed, the model computes a new estimate of A on the updated row based on values obtained during the previous iteration.

This series of operations is performed for each row in the subdivided *ir* grid block, until the end of the grid defined in *grid* is reached. Control is then returned to *model*, which passes on to the next (ir+1) reference grid block.

8. *ctrida*: Utility routine which is called by *fdcalc* to perform the double sweep elimination to solve the implicit set of equations.

9. *diss*: Called by *con*. *diss* calculates frictional dissipation coefficients based on values of the switches read in by *inref*.

10. *wvnum* : Called by *model*, *grid* and *con*. *wvnum* performs a Newton-Raphson solution of the linear wave-current dispersion relation to obtain values of the wavenumber k.

11. *rand*: Called by *model*. This function is a simple random number generator used to initialize the random wave phases if the directional spreading model is being used.

12. *acalc*: Called by *model*. *acalc* normalizes the directional spectrum energy density over a 90° sector.

13. *bnum*: Called by *acalc*. *bnum* computes the Bernoulli number $n!/k!(n-k)!$

14. *fact*: Called by *bnum*. *fact* computes the factorial $n!$ of an integer n.

15. *infile*: Called by *inref*. *infile* provides information needed to define the *namelist* input file. For the standard version of the program, the required code is obtained by including the file *infile1.f*, and the file name is automatically set to *indat.dat*. For the *LRSS* version of the code, where all file names have to be freely definable, the required code is obtained by including *infile2.f*, which establishes the code needed to read the file name from the command line. This latter option requires the library *liblrss.a*.

2.4 Special Installation Instructions

Several features of the program **REF/DIF 1** may require some modification or customizing during program installation on various systems. **REF/DIF 1** is written using the features of FORTRAN 77. No use is made of vectorized solution techniques, so the program should be useable on a wide range of systems with little or no modification.

If the program is to be used on machines with no upward limit placed on the size of the compiled, executable code, only one variable has to be checked during the initial installation of the program. This is the logical device number for the controlling input date file. This number may vary from system to system. It appears in the program as:

iun(5): logical device number for controlling input data file. (initialized near the top of subroutine *inref*).

The supplied version of **REF/DIF 1** has this value set to *iun*(5)=5. All output is directed to data files with pre-chosen unit numbers.

Many systems require that access to disk files be initialized and terminated by the use of **open** and **close** statements in the program code. Since the parameter list of the **open** statement varies from system to system, the user should take care that the **open** statements are compatible with the system software being used. The **open** statements appear near the top of *refdif1* and *inref*. The corresponding **close** statements appear near the end of *refdif1*.

2.5 Computational Grids and Grid Interpolation

The reference grid terminology is defined in Figure 3. The grid consists of a mesh of points with dimensions *mr* x *nr* in x and y. The values of *mr* and *nr* must be less than or equal to the parameters *ixr* and *iyr* whose values are set in the **parameter** statement in *param.h*. Reference grid data of depth *dr* and ambient current components *ur* and *vr* are defined at the grid points. The program assumes that the x, y coordinate system is established with the origin at grid point (*ir*,*jr*)=(1,1). In this manual, we make the distinction between the terms "grid row " *ir*, which is the row of points *jr = 1, nr* at location *ir*, and "grid block" *ir*, which is the physical space between grid rows *ir* and *ir*+1.

The reference points are separated by spacings *dxr* and *dyr* which are uniform in the x and y directions. The spacings *dxr* and *dyr* may have arbitrary, independent values.

Values of *dr*, *ur* and *vr* constitute the "reference grid data". Section 2.8 describes the required input data file for these quantities.

The computational grid for any particular application should be chosen with care. Since **REF/DIF 1** tries to use at least 5 points per wavelength, the length of the computational domain in the propagation direction is restricted (with the given **parameter** statements)

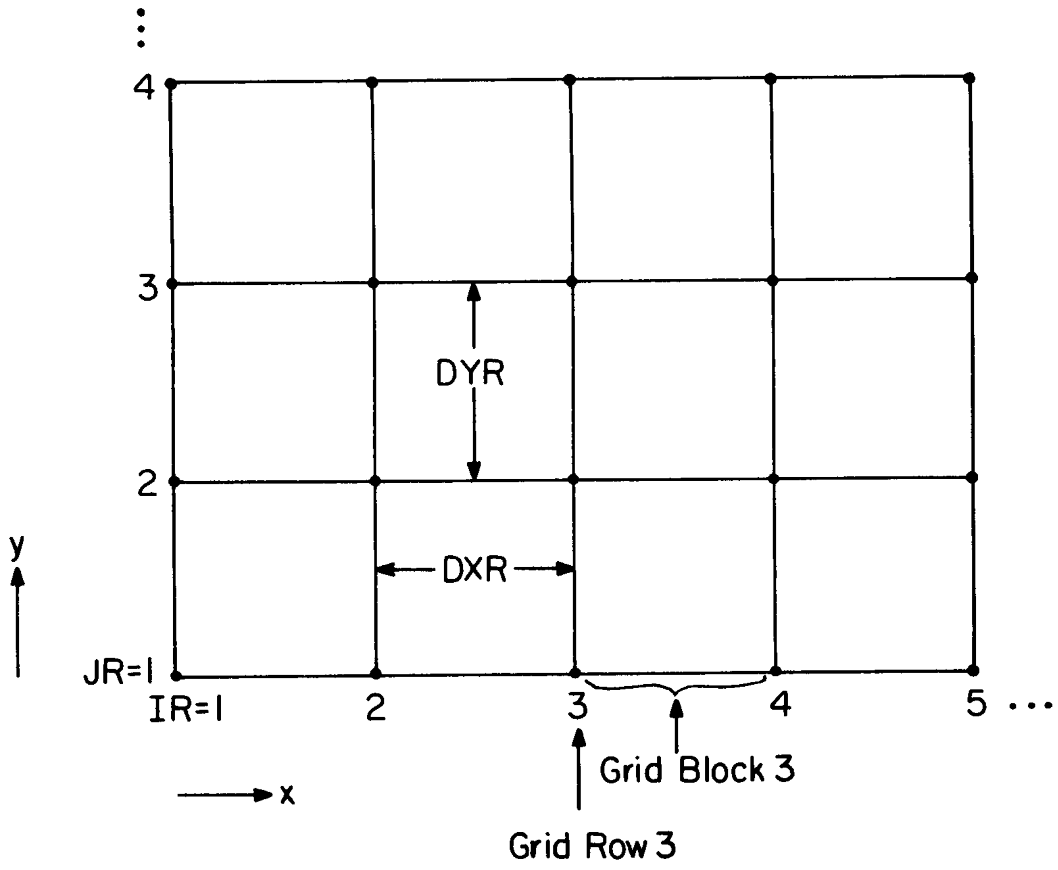

Figure 3: Reference grid notation.

to just over 200 wavelengths. This range can be extended by increasing *ixr* and/or *ix* in the **parameter** statements. The width of the model domain should be chosen such that interference from the boundaries does not affect the study area. If the reflective boundary conditions are used, the extent of boundary influence is usually obvious, particularly when viewing 2-dimensional plots of the amplitude envelope $|A(x, y)|$.

2.5.1 Grid Subdivision

The only major feature of **REF/DIF 1** which is not described in Chapter 1 is the ability to subdivide the given reference grid into a more finely subdivided computational grid. This would usually be done in cases where the reference grid spacing is too large to be used directly for calculations. In this case, the user may specify how the reference grid is to be subdivided, or the user may tell the program to attempt its own subdivisions.

The maximum reference grid dimensions have been set fairly arbitrarily and may be changed by modifying the parameter statements at the beginning of each routine in the program. The grid is large enough to comply with typical grids for various tidal computational models, which may be used to specify the ambient currents, and get small enough so that data values do not occupy too much internal storage. It is anticipated that the spacings *dxr* and *dyr*, if based on such a model, may be too large for an accurate integration of the parabolic model to be performed. Consequently, the model has been arranged so that the reference grid block between any two adjacent reference grid rows may be subdivided down to a finer mesh in order to provide sufficient resolution in the computational scheme. This subdivision process is performed internally in the program (with an exceptional feature to be described below), and may be controlled by the user or the program.

Remember that the computational scheme proceeds by marching in the x-direction, and, therefore, the only reference grid information required at any particular step is the data on row *ir*, where computation starts, and on row *ir*+1, where computation ends. The fine, subdivided mesh is set up on the intervening grid block. An example of a particular subdivision of a grid block is shown in Figure 4. Here, the choices of x-direction subdivision *md*(*ir*)=5 and *nd*=2 are illustrated, with *md* and *nd* being the number of spaces each reference grid cell is divided into, rather than the number of extra points being inserted.

Several restrictions are placed on the choice of *nd* and *md*'s. First, the maximum dimensions of the subdivided grid cell is given by the parameters *ix* and *iy*. This implies that any *md* can be at most (*ix*-1) and *nd* can be at most *(iy-1)/(nr-1)*. The maximum number of added spaces may be increased by increasing *ix* and *iy* in the **parameter** statements. Further, the y-subdivision specified by *nd* is applied uniformly along each grid row for the full extent to the reference grid. No provision is made for variable grid spacing in the

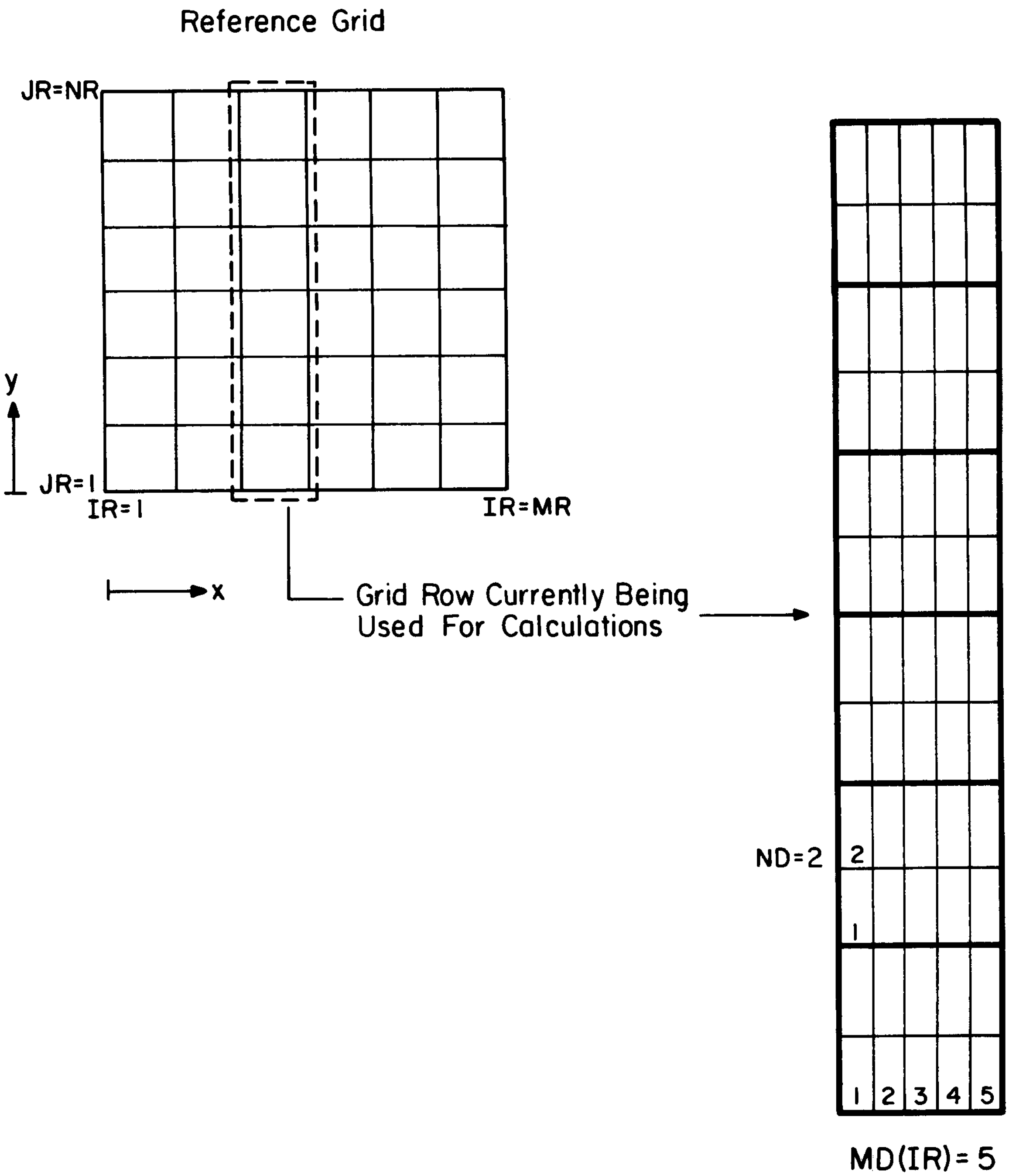

Figure 4: Sample grid subdivision

y-direction. Grid spacing in the x-direction is arbitrary, so *md*'s may differ arbitrarily for each grid block.

The user must specify the single value of *nd* in the input data. Two choices may be made regarding *md*'s, however.

1. The user may let the program calculate *md*. The program proceeds by calculating an average wavenumber along the reference grid row, and uses this to estimate the wavelength in the grid block. The program then chooses a subdivision so that at least 5 grid points per wavelength will occur in the grid block. The program checks to see that this desired number of subdivisions does not exceed the maximum. If it does, the program reduces the number to the maximum and prints a warning message indicating that the grid block can not be subdivided finely enough. Computed results in this case must be regarded as being suspect. The number of subdivisions used is indicated on the output. The user chooses this option by setting the switch *ispace*=0 on input.

2. The user may specify each value of *md*(*ir*) from *ir*=1 to (*mr*-1). This is done by setting the switch *ispace*=1 on input, and the values of *md* are then read in from the input data file. Note that this choice is necessary if the user-defined subgrids discussed below are to be used, since the user will be inputting subgrids with pre-specified spacings. As it is presently written, the program will only print a warning if it encounters subgrids with *ispace*=0 chosen; extensive garbling of input data may result.

After the subdivided grid block is established, the program uses this grid as the actual computation grid. New values of depth d and ambient current u and v are calculated at the extra grid points by fitting a twisted surface to the reference grid using linear interpolation. An example of the resulting bottom topography for a single grid cell is shown in Figure 5.

2.5.2 User-specified Subgrids

In some applications, an important topographic feature may be present at a subgrid scale within the reference grid. Examples include artificial islands, shoals, borrow pits, etc., which are superimposed on an otherwise slowly-varying topography which is represented by the sort of grid resolution appropriate to tidal models. An illustration of such a feature is shown in Figure 6, where a poorly resolved feature occupies portions of four reference grid cells. For cases such as this, the program includes the option for the user to input user-defined, subdivided grid cells in order to specify these features at the level of the computational grid.

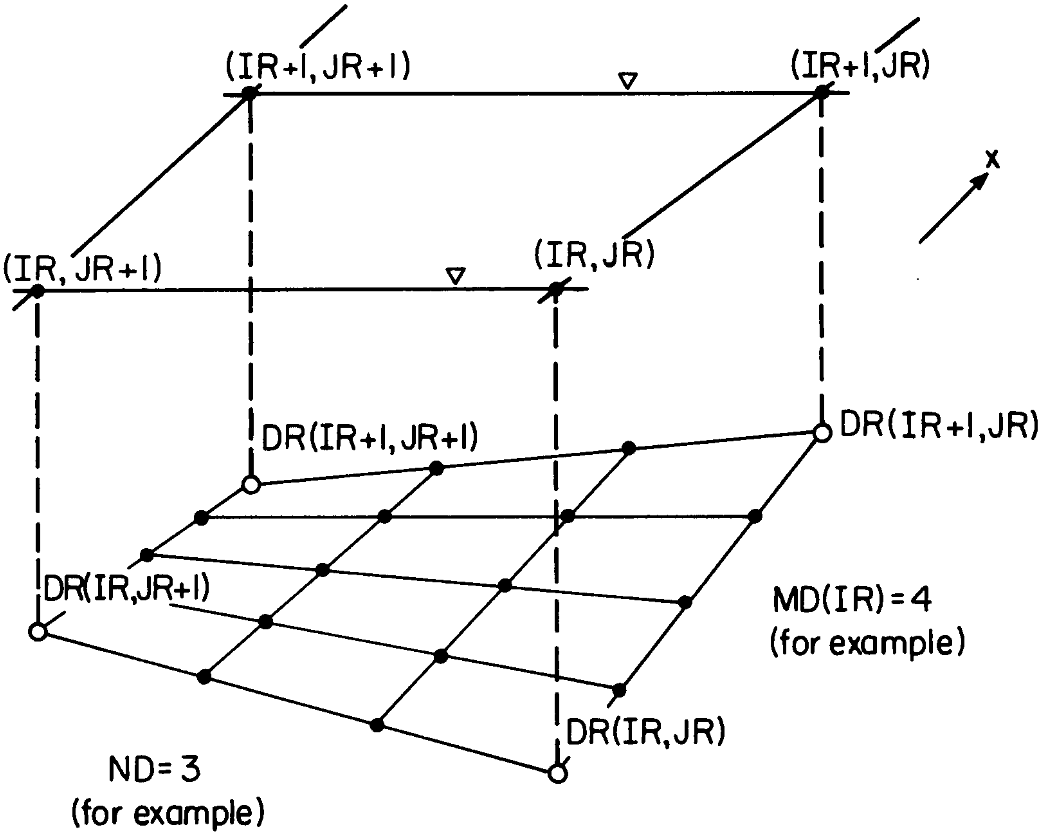

Figure 5: Interpolation of depth data

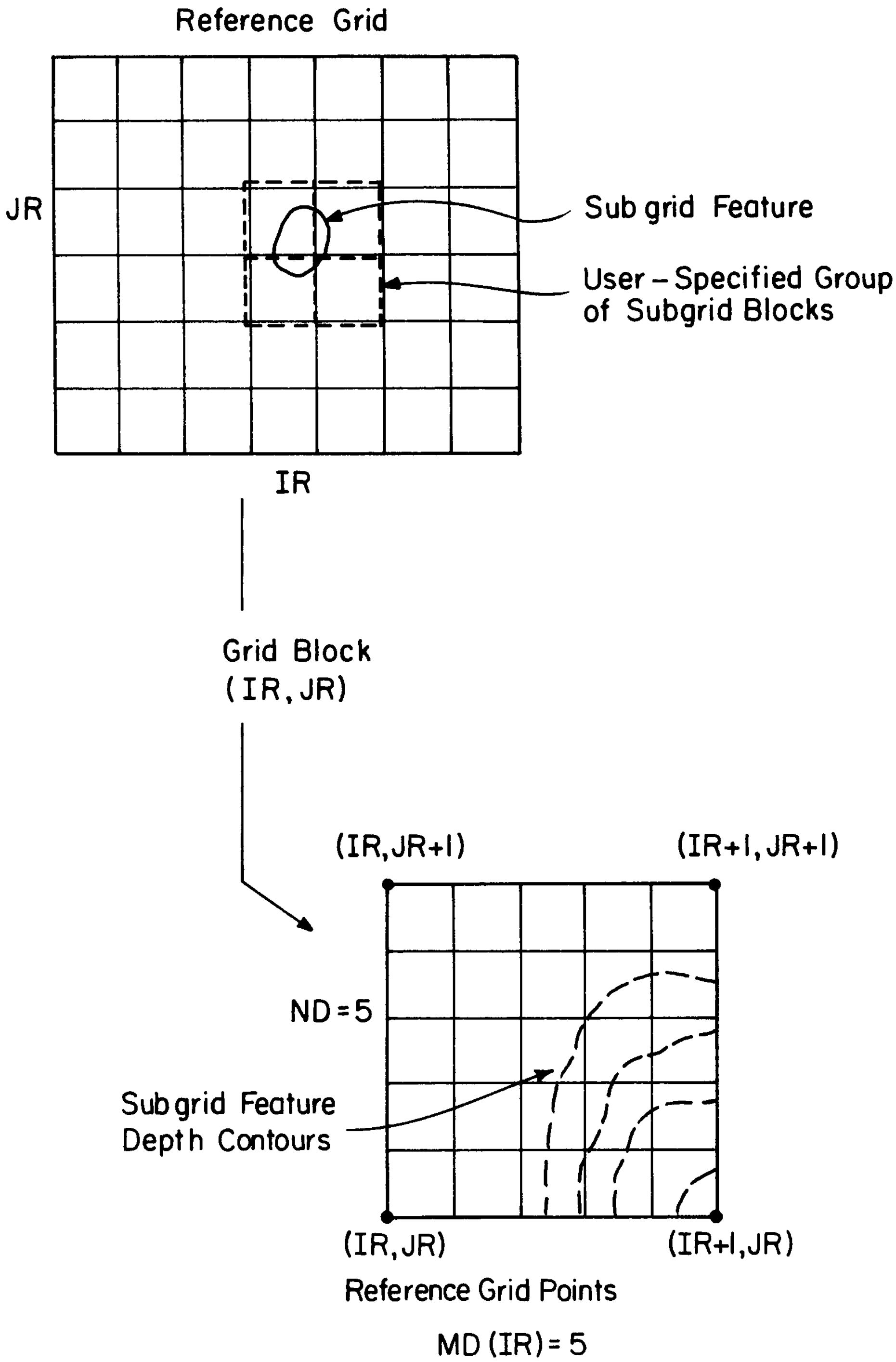

Figure 6: User-defined subgrids

The use of user-defined subgrids implies that the user will be choosing the grid spacing of the subgrids. Since grid spacings must be uniform across a grid block, the user should choose the option, *ispace*=1 and specify the values of *md* in the input file, *indat.dat*. Then, the program will look for a data array of the correct, pre-specified dimension when it reads the subgrid data from the file, *subdat.dat* (unit number *iun*(2)).

Several aspects of the subgrid data should be noted. Data for depth d and current velocities u and v need to be specified at each of the subgrid points. These data are input in the same units, and with the same datum for depth as the data for the reference grid. Also, note that the subgrid includes the points on its boundaries; for example, the single subgrid shown in Figure 6 has a dimension of 6 by 6.

The data values on the outer borders of the subgrids should be setup to match with the linearly interpolated values in the region external to the user-defined subgrid. If the data do not vary linearly along the subgrid boundaries, there may be some mismatch between the subgrid boundaries and the external region. An exception to this rule occurs when two subgrids adjoin each other. Then, care should be taken that the data along the common boundary match. These common boundary point data are duplicated in the input data, since each subgrid data set includes the boundaries.

Instructions and formats for creating the subgrid data file *subdat.dat* are included in section 2.8.

2.6 User Specification of Complex Amplitude on First Grid Row

This section discusses the option of inputing the values of complex amplitude A for the first grid row from an external data file. This option is invoked by setting the data value *iinput*=2 in line 8 of *indat.dat*. (See the following section). In the event that the option is chosen, the data should be stored in a file *wave.dat*. The format for writing data to the file should be similar to:

```
complex a(iy)

write(*,*)(a(j),j=1,n)
```

The data will be read in by the *model* subroutine. The data is read in <u>after</u> grid subdivision in the y direction, and hence n rather than nr data values need to appear in the file *wave.dat*. An insufficent number of data points will simply trigger an "end of file" type of error during the **read** process.

The user may have any number of reasons for wanting to compute A externally to the program. Several possibilities are:

- Running tests with complex initial conditions (as in waves through a breakwater gap, etc).
- Testing a directional distribution model which is different from the model chosen here.
- Specifying a nearly planar wave field which has height and angle variations along the y direction.

To serve as an example, consider the case where you wish to specify a wave field having varying height $2a(y)$ and direction $\theta(y)$ along an offshore boundary having nonuniform depth $h(y)$. Assuming that the local wavenumber $k(y)$ has been determined from the dispersion relation

$$\frac{2\pi}{T} = (gk \tanh kh)^{1/2} \tag{29}$$

where T is the wave period, then the complex amplitude A can be specified by

$$A(y) = a(y)\mathrm{e}^{i\psi(y)} \tag{30}$$

where the phase function $\psi(y)$ is given by

$$\psi(y) = \int_{y=0}^{y} k(y) \sin\theta(y) \mathrm{d}y \tag{31}$$

The discrete values $A(j)$ are then simply given by

$$A(j) = A(y_j); j = 1, \cdots, n \tag{32}$$

Note that this option is not available to users of *REF/DIF 1* in the *LRSS* system.

2.7 Program Input: Model Control and Wave Data

This section discusses the structure of the input data file *indat.dat*, which is read in through logical device number *iun*(5). This file contains all the information needed to control the operations of the program, and all of the input wave data. Reference grid values of depth *dr* and currents *ur* and *vr* are treated in the following section.

Data is read from *iun*(5) in both the *inref* and *inwave* subroutines. The data is arranged in several lists and put in the *indat.dat* file using the *namelist* convention. The various namelist groups are defined according to the following prototype *namelist* statement.

```
      namelist /ingrid/ mr, nr, iu, ntype, icur, ibc, dxr, dyr, dt,
     1                  ispace, nd, iff, isp, iinput, ioutput
     1         /inmd/  md
     1         /fnames/ fname1,fname2,fname3,fname4,fname5,fname6,
     1                  fname7,fname8,fname9,fname10,fname11,fname12,
     1                  fname13,fname14,fname15
     1         /waves1a/ iwave, nfreqs
     1         /waves1b/ freqs, tide, nwavs, amp, dir
     1         /waves1c/ thet0, freqs, tide, edens, nwavs, nseed
     1         /waves2/ freqin, tidein
```

The definition of each input variable follows.

ingrid **namelist group**

- *mr*, *nr* :

  Reference grid dimensions. Maximum values are *ixr*, *iyr* respectively.

- :

  iu is switch for physical units; *iu*=1, MKS; *iu*=2, English. Program defaults to *iu*=1 if an error is made on input.

- *ntype*:

  ntype is switch for nonlinearity; *ntype*=0, linear model; *ntype*=1, composite model; *ntype*=2, Stokes wave model.

- *icur*:

  icur is switch for input current data. *icur*=0, no currents input; *icur*=1, currents input.

  icur defaults to a value of zero if an input error is detected.

- *ibc*:

  ibc is the boundary condition switch. *ibc*=0, closed boundaries; *ibc*=1, open boundaries.

ibc defaults to a value of zero if an input error is detected.

- *dxr, dyr* :

  Reference grid x-spacing and y-spacing, which are assumed to be uniform over the entire reference grid.

- *dt*:

  Depth tolerance value.

- *ispace, nd*:

  ispace is switch controlling subdivisions; *ispace*=0, program attempts its own x subdivisions. *ispace*=1, user specifies x subdivisions.

- *nd*:

  nd is the number of y-direction subdivisions (needed in either case).

- *iff(1), iff(2), iff(3)*:

  Dissipation switches; iff(1)=1;turn on turbulent boundary layer; iff(2)=1; turn on porous bottom damping; iff(3)=1; turn on laminar boundary layers. No damping if all values are zero.

- *isp*:

  Switch for user-specified sub-grid specifications. *isp*=0, no subgrids to be read; *isp*=1, subgrids will be read.

- *iinput*:

  iinput specifies whether the program or the user will generate the first row of complex amplitude A values. If *iinput*=1, the program constructs A based on the input conditions specified as follows. If *iinput*=2, the user must specify A in an external data file *wave.dat*. This option is not available in *LRSS*

- *ioutput*:

  ioutput specifies whether the last computed row of complex amplitudes A are to be stored in file *owave.dat*. A value of *ioutput*=1 skips this option. *ioutput*=2 turns the option on.

inmd **namelist group**

- *md(ixr)*:

  If *ispace*=1, the x-direction subdivisions are inserted here (one for each reference grid block). The array *md* must be dimensioned according to the value of *ixr* in the main program, regardless of how many values actually are being used.

fnames **namelist group**

- *fname1*: *refdat.dat*, reference grid data file.
- *fname2*: *outdat.dat*, old standard output data file. (This file will be dropped in some intermediate Version 2.5 revisions.)
- *fname3*: *subdat.dat*, user-specified subgrids.
- *fname4*: *wave.dat*, user-specified complex amplitude on row 1 (for *iinput* = 2).
- *fname5*: *owave.dat*, complex amplitude on last row (for *ioutput* = 2).
- *fname6*: *surface.dat*, complex amplitude data for constructing an image of the instantaneous water surface at the computational resolution. If *REF/DIF 1* is given a null string as the input for this file name, no file is generated.
- *fname7*: *bottomu.dat*, magnitude of bottom velocity at reference grid points. If *REF/DIF 1* is given a null string as the input for this file name, no file is generated.
- *fname8*: *angle.dat*, wave directions θ in degrees at reference grid points. This file is always generated.
- *fname9*: not used at present, supply dummy name.
- *fname10*: *refdif1.log*, run log for *refdif1* program.
- *fname11*: *height.dat*, wave heights at reference grid locations. This file is always generated.
- *fname12*: *sxx.dat*, S_{xx} components at reference grid locations. If *REF/DIF 1* is given a null string as the input for this file name, no file is generated.
- *fname13*: *sxy.dat*, S_{xy} components at reference grid locations. If *REF/DIF 1* is given a null string as the input for this file name, no file is generated.
- *fname14*: *syy.dat*, S_{yy} components at reference grid locations. If *REF/DIF 1* is given a null string as the input for this file name, no file is generated.
- *fname15*: = —depth.dat—, tide-corrected depths at reference grid locations. This file is always generated.

If *iinput*=1, the rest of *indat.dat* is as follows:

waves1a **namelist group**

- *iwave*:

  iwave is switch for wave field type. *iwave*=1, discrete wave components. *iwave*=2, directional spreading model.

- *nfreqs*:

  nfreqs is the number of frequency components to be run. Maximum is value of *ncomp* in *param.h*.

The remainder of the file depends on the choice of *iwave*.

For *iwave*=1:

waves1b **namelist group**

- *freqs(ncomp)*:

  Wave period for each frequency component. *ncomp* values must be given.

- *tide(ncomp)*:

  Tidal offset for each frequency component. *ncomp* values must be given.

- *nwavs(ncomp)*:

  Number of wave components for each frequency. *ncomp* values must be given.

- *amp(ncomp,ncomp)*:

  Amplitude (not height) for each component wave.

- *dir(ncomp,ncomp)*:

  Direction in degrees relative to x axis for each wave component.

For *iwave*=2:

waves1c **namelist group**

- *thet0*:

  The central direction for the model spectrum.

- *freqs(ncomp), tide(ncomp)*:

  As above.

- *edens(ncomp)*:

  Variance density (m^2 or ft^2) for each frequency component.

- *nwavs(ncomp)*:

  Directional spreading factor (the factor n in $\cos^{2n}(\theta/2)$.

- *nseed*:

  The seed value for the random number generator (between 0 and 9999).

For *iinput*=2, the remainder of the data file is:

waves2 **namelist group**

- *freqin, tidein*:

  Wave period and tidal offset for the single frequency component.

Examples of *indat.dat* data files are given in the example problem section. Several things must be kept in mind while constructing a *namelist* - oriented data file. *namelist* does not allow a part of an array to be read or written. It is therefore imperative that the number of values being entered into any dimensioned variable in a *namelist* group be equal to the dimension of that variable in **REF/DIF 1**. It is thus highly recommended that any program being used to construct the *indat.dat* file should use the *param.h* file to specify parameters. In any event, the user should consult the *datgenv25.f* or *indat-convertv25.f* programs to get a feel for how the file is constructed. The file may also be easily constructed by hand.

In constructing the *indat.dat* in the provided programs, we have followed the most restrictive convention that was found, which is that the items in a namelist group must be read in in the order in which they are specified in the *namelist* statement. This restriction is imposed by the Silicon Graphics Fortran compiler. Most compilers will allow an arbitary ordering of variables within each namelist group.

2.8 Program Input: Reference Grid and Subgrid Data

This section describes two files which provide arrays of data on the various grids.

Reference Grid Data File

This file (named *refdat.dat*) is accessed as logical device number *iun*(1). Its contents consist of the arrays of depth *dr*, x-velocity *ur*, and y-velocity *vr* at the reference grid points. This file is accessed only once per model run, and its entire contents are read in by subroutine *inref.* If $icur = 0$, only the depth data *dr* need to be specified.

Data for this file should be written in the following format:

```
        do 1 ir=1, mr
        write(unit,#) (dr(ir, jr), jr=1, nr)
1       continue

        (then, if icur=1)

        do 2 ir=1, mr
        write(unit,#) (ur(ir, jr), jr=1, nr)
2       continue

        do 3 ir=1, mr
        write(unit,#) (vr(ir, jr), jr=1, nr)
3       continue
#       format(20f10.4)
```

The data may be in either MKS or English units; set the units switch *iu* in the *iun*(5) data file *indat.dat* accordingly.

Subgrid Data File

This file (named *subdat.dat*) is accessed as logical device number *iun*(2). If no user-defined subgrids are to be read in, this file may be omitted. The file consists of two parts; an integer array of 1's and 0's indicating which reference grid cells are to be defined by the user, and then a sequence of groups of arrays of *d*, *u* and *v*, one group for each subgrid.

The integer array *isd* is dimensioned (*mr*-1) by (*nr*-1), with one point for each spacing in the reference grid. The array contains a 0 if that cell is not to be user-defined, and a one if it is. For example, the (*mr*,*nr*)=(7,6) reference grid shown in Figure 6 has four cells which are to be read in as user-defined subgrids.

The array *isd* should be written in the data file first, using the format:

```
      do 1 ir=1,mr-1
      write(unit,#)(isd(ir,jr),jr=1,nr-1)
1     continue
#     format(15I4)
```

Now, a group of arrays of d, u and v must be entered into the data file, with one group corresponding to each value of 1 in *isd*. The program accesses the data file by x-row (*ir*) and then by y-column (*jr*). For the above example, the subgrid array groups should thus be stored in the order of the coordinate pairs (4,3), (4,4), (5,3), (5,4). The dimensions of each subgrid are given by $m=md(ir)+1$ and $ns=nd+1$. The borders of adjacent subgrids share the same common boundary points.

Each group of subgrid data should be written using a format similar to:

```
      write(unit,#)((d(i,j),j=1,ns),i=1,m)

      (then, if icur=1)

      write(unit,#)((u(i,j),j=1,ns),i=1,m)
      write(unit,#)((v(i,j),j=1,ns),i=1,m)

#     format(20f10.4)
```

using the appropriate value of m for the grid block in question. Data may be in MKS or English units, depending on the value of *iu* in the input data file. The integer array *isd* is read by *inref*, and the individual subgrids are read by *grid*.

2.9 Program Output

This section discusses output of two forms: output sent to a log file, and array output stored in disk files.

2.9.1 Output log file

Log file output consists of three types: title and header information which is printed in order to identify the run and the operating perameters, runtime error messages (either warnings or terminal error messages), and information about calculations on each grid row. The default name for the file is *refdif1.log*.

Header Information

At the start of each run, a set of two title pages are printed. Page 1 identifies the program and then prints out messages identifying the parameters which set up the model run, as read in by subroutine *inref*. A sample title page 1 is shown in Figure 7.

Title page 2 gives the input wave conditions which were read in by *inwave*. A sample title page 2 is given in Figure 8.

Dimensional quantities are printed on the title page in the units used for input. Title page 1 gives an indication whether quantities are in MKS or English units.

Run-time Error Messages

REF/DIF 1 performs some data checking and checking of calculations during a run. This checking may result in warnings or terminal errors which are beyond calculation errors which would lead to standard FORTRAN error messages. A list of possible errors and the resulting messages follow.

1. Reference grid dimensions were specified as being too large on input. $mr > ixr$ and/or $nr > iyr$.

   Message: dimensions for reference grid too large; stopping.

   Action: Program stops.

   Error occurs in: *inref*

2. User specifies a y-direction subdivision nd which will cause the number of y grid points n to exceed the maximum iy.

   Message: y-direction subdivision too fine. maximum number of y grid points will be exceeded. Execution terminating.

   Action: Program stops.

   Error occurs in: *inref*

3. User specifies an x-direction subdivision on one of the grid blocks ir which exceeds the maximum amount (ix-1). As a result, the dimension of the subdivided grid will be too large.

   Message: x-direction subdivision too fine on grid block "*ir*", execution terminating.

   Action: program stops

   Error occurs in: *inref*

```
                    Refraction-Diffraction Model for
                    Weakly Nonlinear Surface Water Waves

                    REF/DIF 1,  Version 2.5

                    Center for Applied Coastal Research
                    Department of Civil Engineering
                    University of Delaware
                    Newark, Delaware 19716

          James T. Kirby and Robert A. Dalrymple, November 1994
0
                    input section, reference grid values

 reference grid dimensions  mr=100
                            nr=100

 reference grid spacings   dxr=  5.0000
                           dyr=  5.0000

 physical unit switch iu=1,  input in mks units

 icur=1, current values read from data files

 ibc=0, closed (reflective) lateral boundaries

 ispace =0 chosen, program will attempt its own reference grid subdivisions

 y-direction subdivision according to nd=  1

 ntype = 1, stokes model matched to hedges model

   switches for dissipation terms

 0   turbulent boundary layer
 0   porous bottom
 0   laminar boundary layer

 isp=0, no user defined subgrids
  iinput = 1, program specifies initial row of a
```

Figure 7: Sample title page 1

```
                    input section, wave data values



iwave=1, discrete wave amps and directions


the model is to be run for  1 separate frequency components


frequency component  1

wave period=  8.0000sec., tidal offset=  0.0000

wave component  1, amplitude =  0.5000, direction=  0.0000
```

Figure 8: Sample title page 2

4. A depth value occurs in the reference grid which differs from the average of its neighbors by more than the tolerance value *dt* specified on input. This is basically a data checking feature. Printed values are in meters.

   Message: Depth "*dr*" (m) at reference grid location "*ir*, *jr*" differs from the average of its neighbors by more than "dt" (m). Execution continuing.

   Action: None by program. Data in file *refdat.dat* should be corrected if wrong.

   Error occurs in: *inref*

5. An ambient current value occurs which implies that the flow would be supercritical at the given location. This serves as both a check for anomalously large current values, and an indicator of possible subsequent computational problems.

   Message: ambient current at reference grid location "*ir,jr*" is supercritical with froude number = "froude number", execution continuing

   Action: None by program. Data in file *refdat.dat* should be corrected if wrong.

   Error occurs in: *inref*

6. If the user specifies that predetermined subgrids are to be read in, while at the same time telling the program to performs its own subdivisions, the computed dimensions of the subgrid may be different than those of the subgrid included in the input. Runs requiring user-specified subgrids should choose the *ispace*=1 option. If an incompatible set of dimensions occurs, the program will either garble the input array or run out of data.

   Message: Warning: input specifies that user will be supplying specified subgrids (isp=1), while program has been told to generate its own subgrid spacings (ispace=0). possible incompatibility in any or all subgrid blocks.

   Action: None by program. Should restart unit with correct *ispace*,*isp* values.

   Error occurs in: *inref*

7. While calculating its own subdivision spacings, the model may try to put more division in a reference grid block than is allowed by by dimension *ix*. If this occurs, the program uses the maximum number of subdivisions allowed (*ix*-1), but prints a message indicating that the reference grid spacing is too large with respect to the waves being calculated. This problem may be circumvented by increasing the size of *ix* in *parameter* statements.

   Message: *model* tried to put more spaces than allowed in grid block "*ir*"

Action: Program performs fixup and continues. Model resolution and accuracy may be poor, and a finer reference grid or increased value of *ix* in the *parameter* statements should be used.

Error occurs in: *grid*

8. While using the Stokes wave form of the model, *ntype*=2 the model may encounter large values of the Ursell number, indicating that the water is too shallow for that model to be appropriate. The cutoff point recognized by the program is $(A/h)/(kh)^2 = 0.5$.

   Message: Warning: Ursell number = "u" encountered at grid location "i,j" should be using Stokes-Hedges model (ntype = 1) due to shallow water

   Action: The program should be re-run with the composite nonlinear model.

   Error occurs in: *fdcalc*

9. The Newton-Raphson iteration for wavenumber k may not converge in the specified number of steps. This may occur for waves on strong opposing currents.

   Message: WAVENUMBER FAILED TO CONVERGE ON ROW "I", COLUMN "J"

   K = last iterated value of wavenumber

   D = depth

   T = period calculated from last iterated value of k

   U = x-direction velocity

   F = value of objective function (should be =0 for convergence)

   Action: Program continues with last iterated value of k. Computed results are of questionable accuracy.

   Error occurs in: *wvnum*

Log of Calculations

For each frequency component, the program starts a new page of output and indicates the number of the component in the input stack. The model then prints the x position, the value of the reference phase function and the number of x direction subdivisions used for each reference grid row.

2.9.2 Stored Output

The program stores a set of data files with a resolution of the reference grid points. Each of these files is written using a common format statement and may be read using the format:

```
      do i=1,mr
      read(unit,100)(variable(i,j),j=1,nr)
      end do

100   format(200(f10.4))
```

The available files are:

height.dat: wave height.

depth.dat: water depth with tide correction included.

angle.dat: wave angle in degrees.

sxx.dat, sxy.dat, syy.dat: radiation stresses (not available yet).

bottomu.dat: magnitude of bottom velocity (if requested).

It should be noted that the wave directions given on output are meaningless if multiple direction components are being used, and, for single component runs, become meaningless if the waves become short crested or the crests become significantly curved.

Finally, a file *surface.dat* is generated if *isurface* is set to one. This file provides the same type of information that was put in *outdat.dat* in older versions of the program, with the exception that the present version stores data for every computational point in the domain.

1. n
2. $y(j)$, $j=1$, n

   Then, for each x-position in the computational grid, the program stores the following information.

3. x,
4. $ae^{i\psi}(i,j)$, j=1, n

Since it is not usually known *a priori* how many steps will be taken in the x direction, this file is ended by writing in a negative value of x. This file is processed by the program *surface.f* to give data on a regularly spaced grid.

An anotated listing of the **REF/DIF 1** program code is given in Appendix A. Various preprocessing and postprocessing programs are also listed in Appendix B (for normal usage) and Appendix C (for LRSS usage).

3 EXAMPLE CALCULATIONS

This chapter presents calculations performed using the combined refraction-diffraction model **REF/DIF 1**. The problems studied here were chosen as representative tests of various features of the model. Further examples illustrating the use of the computational schemes upon which the program is based may be found in the technical report by Kirby (1983).

Each section of this chapter describes in full the model's application to a specific problem. Following a description of the problem and an indication of the type of results desired, the input data files for the program are displayed and explained. These data files are then used to run the program **REF/DIF 1** with no job-specific modifications to the program involved. Program output is then presented in such a way as to adequately indicate the results, although, in application, individual users may wish to alter the nature of the presentation of output data.

The output for the various examples has been presented using some plotting programs which are external to the main body of the supplied program **REF/DIF 1**. These specialized programs have been included in order to provide some guidance in reading the data files generated by the main program. However, plotting routines are likely to vary from one computer system to another. The extra programs are therefore likely to be extensively machine-specific to the systems on which the computations were performed.

Section 3.1 presents calculations of waves around an artificial, surface piercing island. This example makes particular use of the breaking wave, thin film, and shallow water dispersion relation capabilities of the model.

Section 3.2 provides calculations for waves propagating over a submerged, elliptic shoal resting on a plane beach. This example has been studied experimentally and provides a means for checking the accuracy of the model calculations. It also provides an example of the type of results provided by a combined refraction-diffraction model in a situation where ray tracing predicts a strong convergence of wave rays, with resulting singularities in the prediction of wave height.

Section 3.3 provides example calculations for the case of waves shoaling on a plane beach and interacting with a rip current. This example illustrates the wave-current interaction feature of the model.

3.1 Waves Around an Artificial Island

The first example involves the calculation of the wave field around an artificial, surface piercing island of the type used in offshore operations. The island is circular with a base radius of $400ft$ and a crest elevation of $80ft$ above the flat seabed. The island radius at the crest is $160ft$, leading to a side slope of $1:3$. The water depth around the island is taken to be $60ft$. The island geometry is shown in Figure 9.

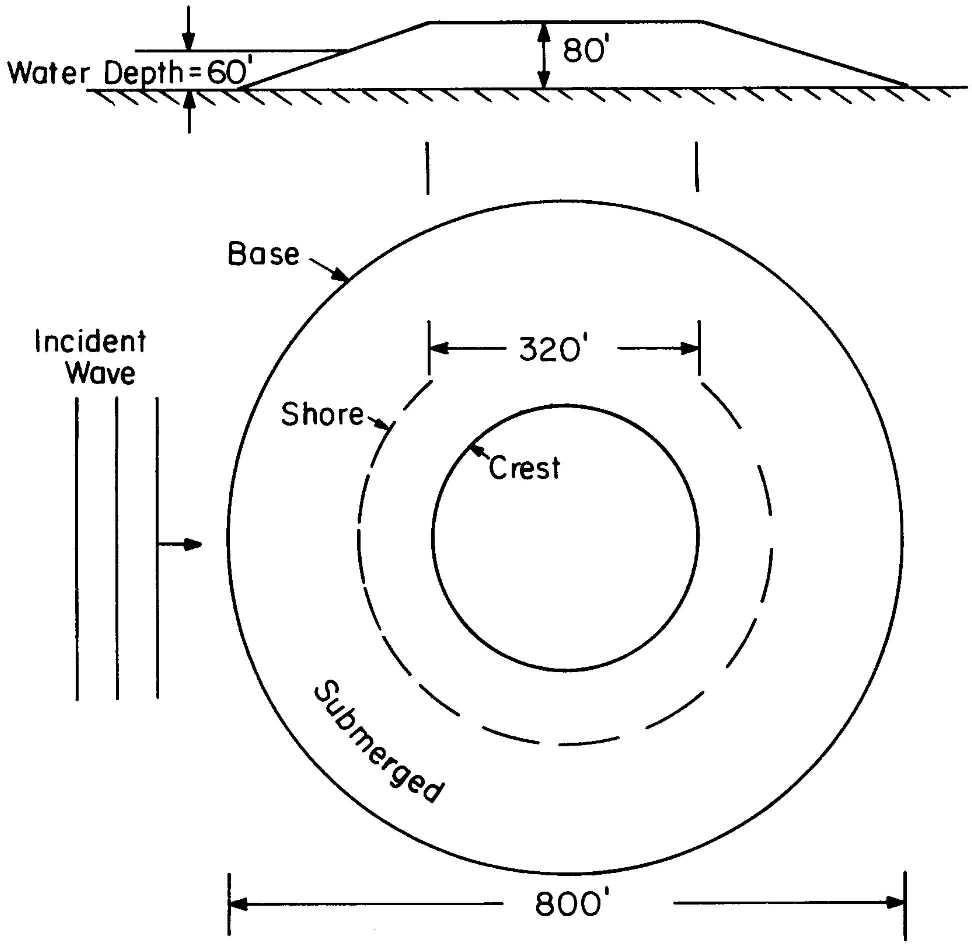

Figure 9: Artificial island geometry

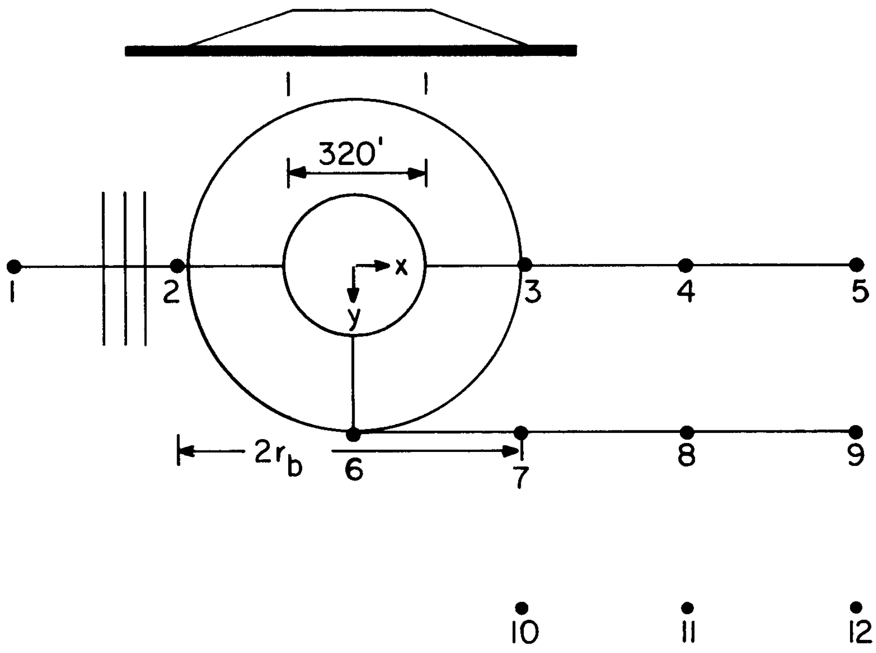

Figure 10: Locations for wave height measurements

The wave conditions to be studied are given by:

- Wave height: $H = 28ft$
- Wave period: $T = 10sec.$
- No currents

The model was run with a depth of $60ft$ away from the island and a tidal offset of $0.0ft$. The required set of wave predictions consisted of wave height at 12 locations as indicated in Figure 10. The spacing between the points are in units of the base radius $r_b = 400ft$. Note that, since the model does not calculate reflected waves, the predicted wave heights at points 1 and 2 will be identical. Therefore, computations may be started arbitrarily close to the leading edge of the island base, in the absence of any current field distorted by the island's presence.

3.1.1 Setting up the Model

First, the island topography was established. Note that the region of the island above the water line is not treated explicitly in the computations. We therefore represented the island

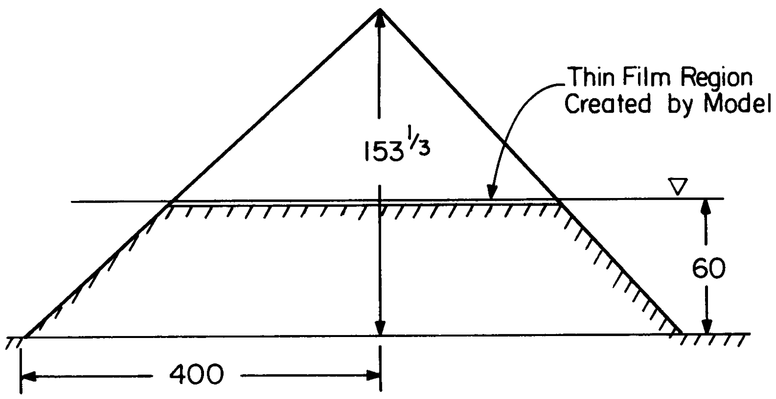

Figure 11: Representation of the island geometry in the program.

in the input data as a right circular cone with a peak height of $153.33ft$ and a base radius of $400ft$. The model will truncate the island in order to create a "thin film" over the exposed portions (see Figure 11).

Next, the reference grid spacing was chosen. Since the physical region to be modelled is small, we picked a reference grid with fine enough resolution so that no subdivision of the reference grid will be required. Using a wave period of 10 seconds and a depth of $60ft$, we use the dispersion relationship

$$\frac{4\pi^2}{T^2} = gk \tanh kh \tag{33}$$

to calculate $kh = .97$, giving a wave length of $L = 389ft$. L is just slightly less than $r_b = 400ft$. The grid spacings dxr and $dyr = 20ft$ were chosen for the reference grid, giving approximately 20 points per wavelength away from the island. We use 100 x 100 storage locations for the reference grid, indicating a model of approximately $5r_b$ by $5r_b$ in x and y. We sited the island center at $x = 460ft$ and $y = 10ft$, where x and y are measured from the computational grid corner. The island and measurement points are shown in relation to the grid in Figure 12.

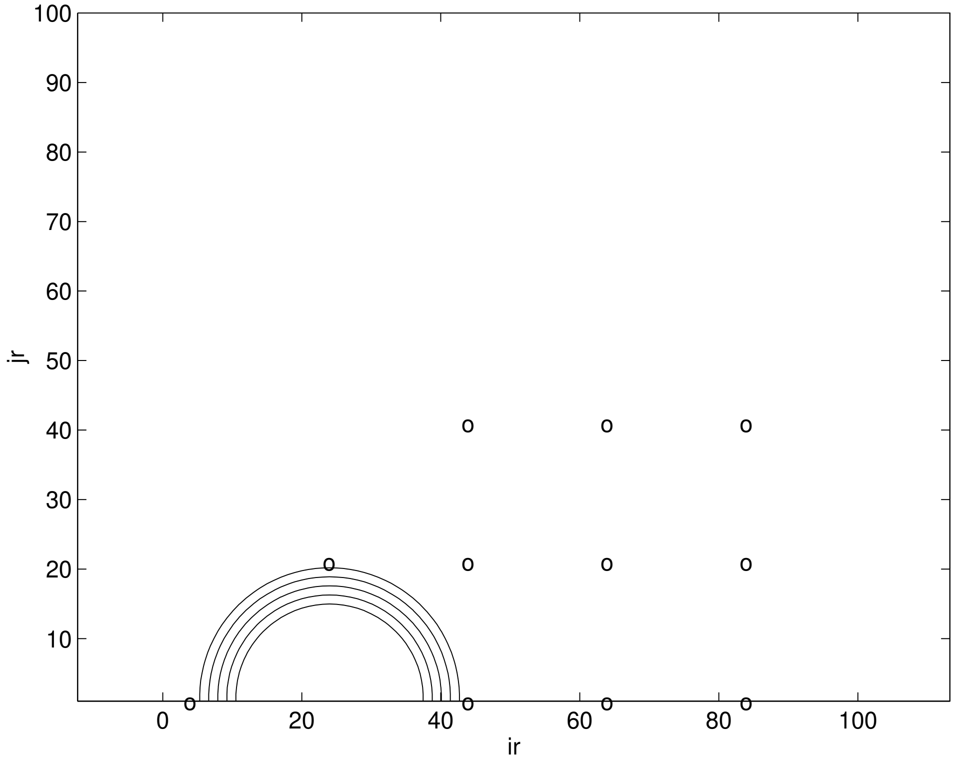

Figure 12: Measurement points in relation to reference grid.

3.1.2 The Input Data Files

One run of the model was performed using the specified input conditions. The input data file *indat.dat* for the run follows. The reference grid data was stored in file *refdat.dat*.

```
$fnames
fname1 =  'refdat.dat '
fname2 =  'outdat.dat '
fname3 =  'subdat.dat '
fname4 =  'wave.dat '
fname5 =  'owave.dat '
fname6 =  'surface.dat '
fname7 =  'bottomu.dat '
fname8 =  'angle.dat '
fname9 =  ' '
fname10 =  'refdif1.log '
fname11 =  'height.dat '
fname12 =  'sxx.dat '
fname13 =  'sxy.dat '
fname14 =  'syy.dat '
fname15 =  'depth.dat '
$end
$ingrid
mr =          100
nr =          100
iu =            2
ntype =            1
icur =           0
ibc =           0
dxr =  20.00000
dyr =  20.00000
dt =  10.00000
ispace =            0
nd =            1
iff =           0,            0,            0
isp =           0
iinput =            1
ioutput =            1
$end
$waves1a
iwave =            1
nfreqs =            1
$end
$waves1b
freqs =  10.00000
tide =  0.0000000E+00
nwavs =            1
amp =  14.00000
dir =  0.0000000E+00
$end
```

3.1.3 Model Results

The output for the artificial island run is presented in two forms. First, Table 1 provides values of wave heights at the measurement locations indicated in Figure 12.

ir	jr	Height (ft)
3	1	28
43	1	17.3
63	1	14.4
83	1	16.9
23	21	23.5
43	21	19.1
63	21	20.9
83	21	18.6
43	41	32.1
63	41	29.7
83	41	23.5

Table 1: Calculated wave heights at measurement locations.

In addition to this, we have constructed contour plots of instantaneous surface elevation and wave height; these are shown in Figures 13 and 14, respectively. Contour elevations for wave height are in increments of 5 ft.

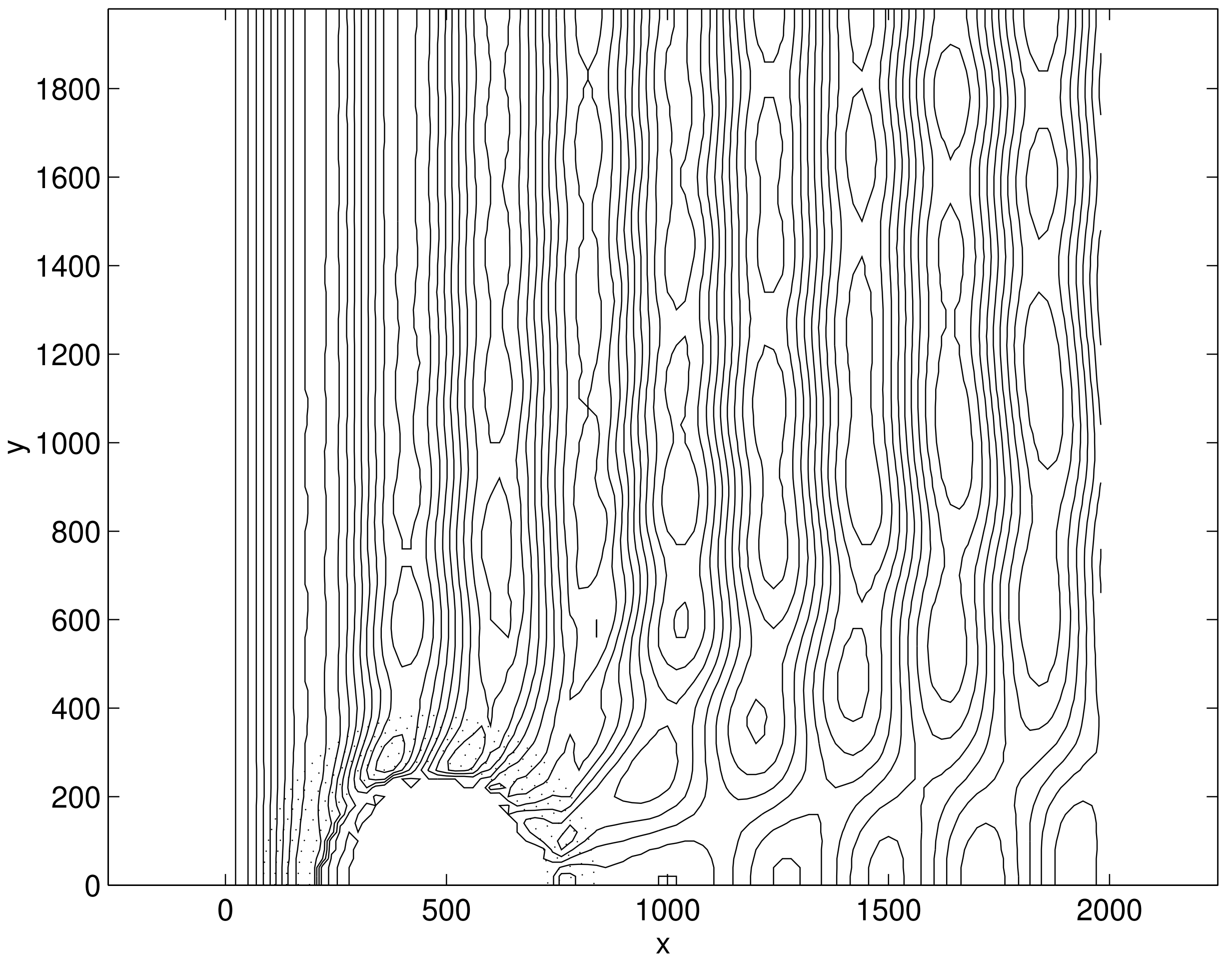

Figure 13: Artificial island example: contours of instantaneous surface elevation.

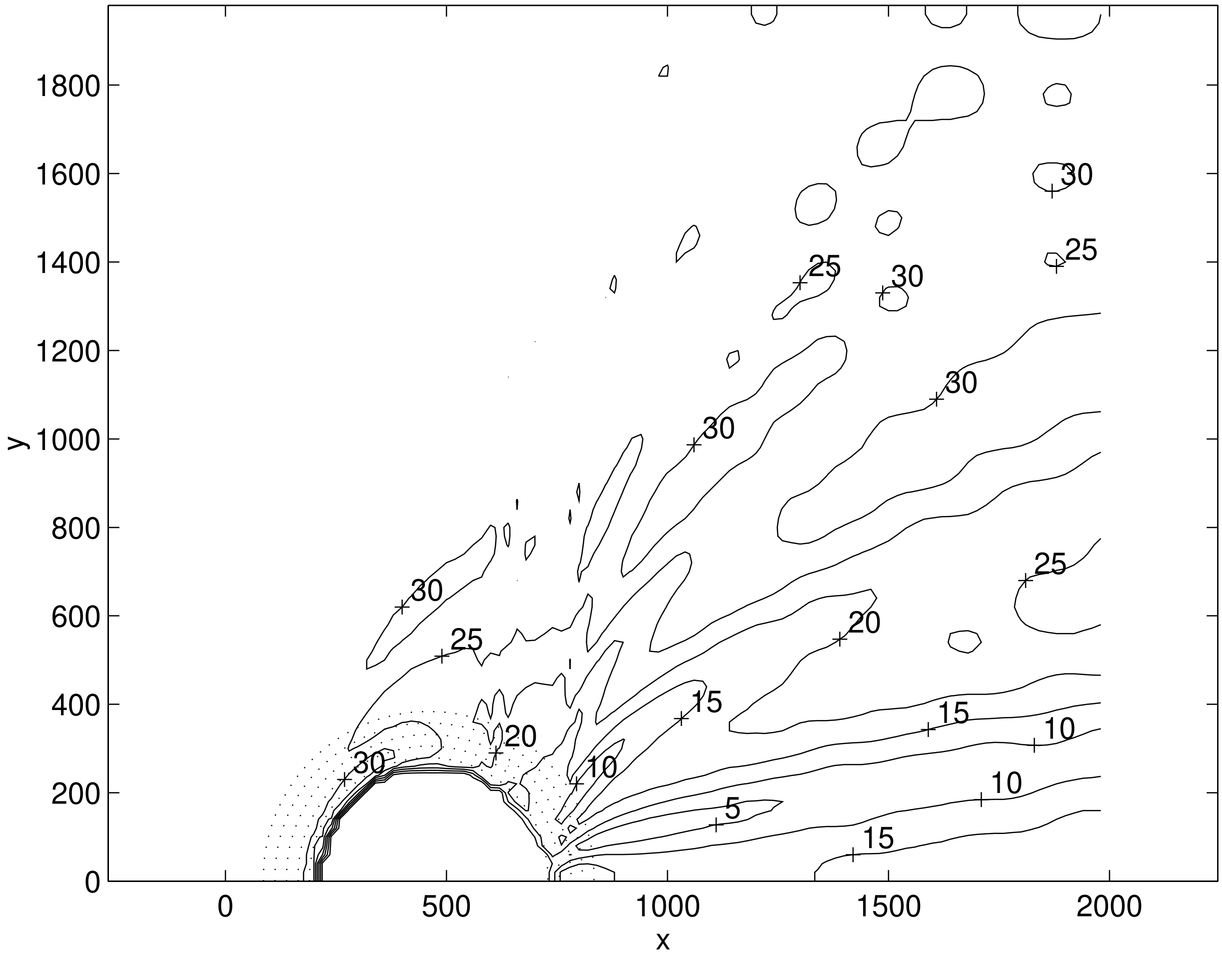

Figure 14: Artificial island example: contours of wave height.

3.2 Wave Focussing by a Submerged Shoal

In this example, we study the propagation of an initially plane wave over a submerged shoal resting on a plane beach. This example has been chosen for several reasons:

1. A carefully controlled set of waves measurements has been made in the laboratory for this case (see Berkhoff et al (1982); Kirby (1986a)).

2. The wave pattern represents the case of ray crossing in the refraction method, and thus the computed results indicate present method's utility in situations where ray tracing breaks down.

3. The example gives a thorough test of the accuracy of the large angle and composite nonlinearity formulations.

The topography to be studied is shown in Figure 15. Details of the calculation of the topography may be found in Kirby (1986a).

For the case of an incident plane wave, we have performed a run of the model using input data corresponding to the experiment of Berkhoff et al (1982). The run was done using the full model with the Stokes-Hedges composite nonlinearity.

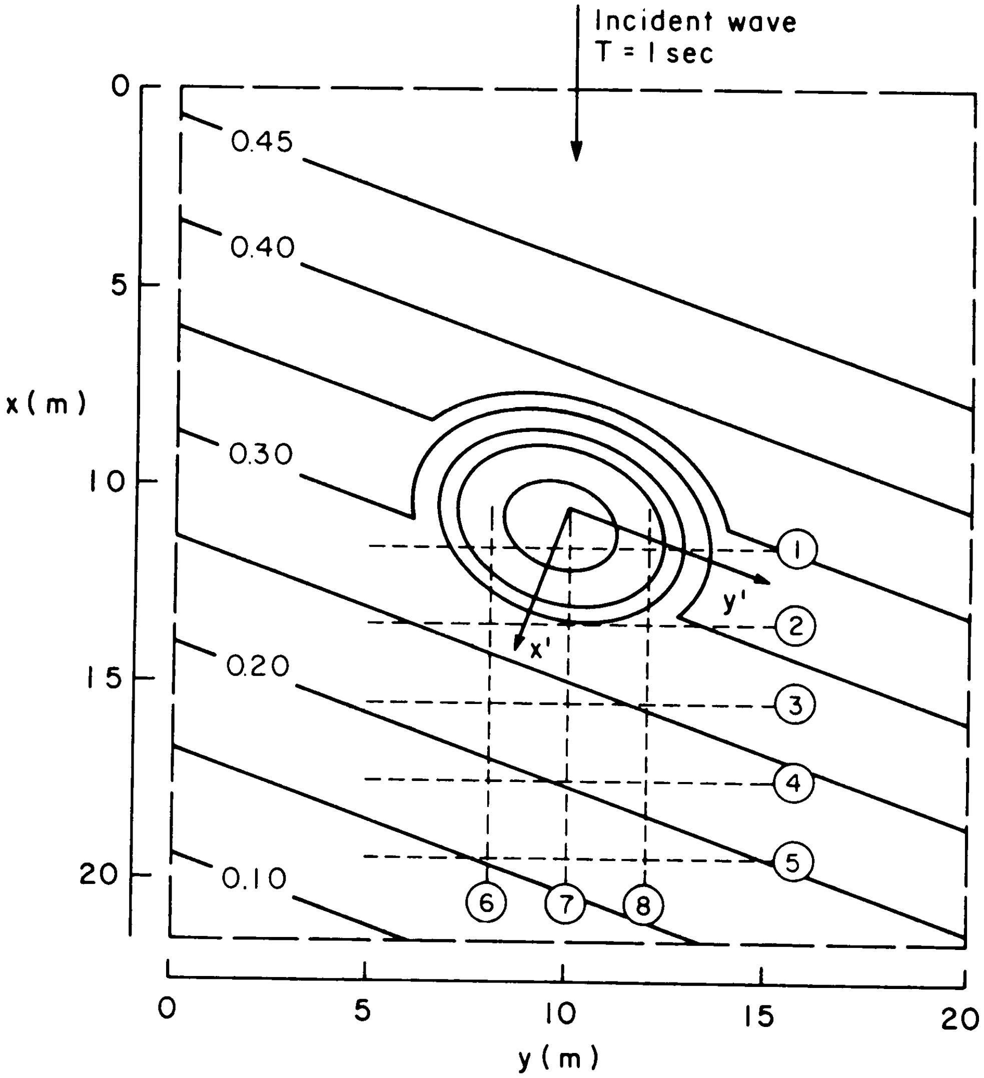

Figure 15: Bottom contours and computational domain for the experiment of Berkhoff et al (1982). Experimental data on transects 1-8.

3.2.1 The Input Data Files

The input data file *indat.dat* for the present case follows.

```
$fnames
```

$\cdots$ same as previous example $\cdots$

```
$end
$ingrid
mr =           100
nr =           100
iu =             1
ntype =             1
icur =            0
ibc =            0
dxr =  0.2500000
dyr =  0.2500000
dt =  10.00000
ispace =             0
nd =             1
iff =            0,             0,             0
isp =            0
iinput =             1
ioutput =             1
$end
$waves1a
iwave =             1
nfreqs =             1
$end
$waves1b
freqs =  1.000000
tide =  0.0000000E+00
nwavs =             1
amp =  2.3200000E-02
dir =  0.0000000E+00
$end
```

The supplied program *datgenv25.f* may be used to generate the depth grid *refdat.dat*.

3.2.2 Model Results

The output files *depth.dat, height.dat, surface.dat* for this example have been used to construct plots of instantaneous surface elevation and wave height. In this case, wave heights have been non-dimensionalized using the incident wave height. The resulting plots are shown in Figures 16 and 17. The plots show the effect of wave focussing over the shoal area, and show that the prediction of the wave field beyond the shoal does not involve the problem of caustic (or singularity) formation common to ray-tracing algorithms used to model similar situations.

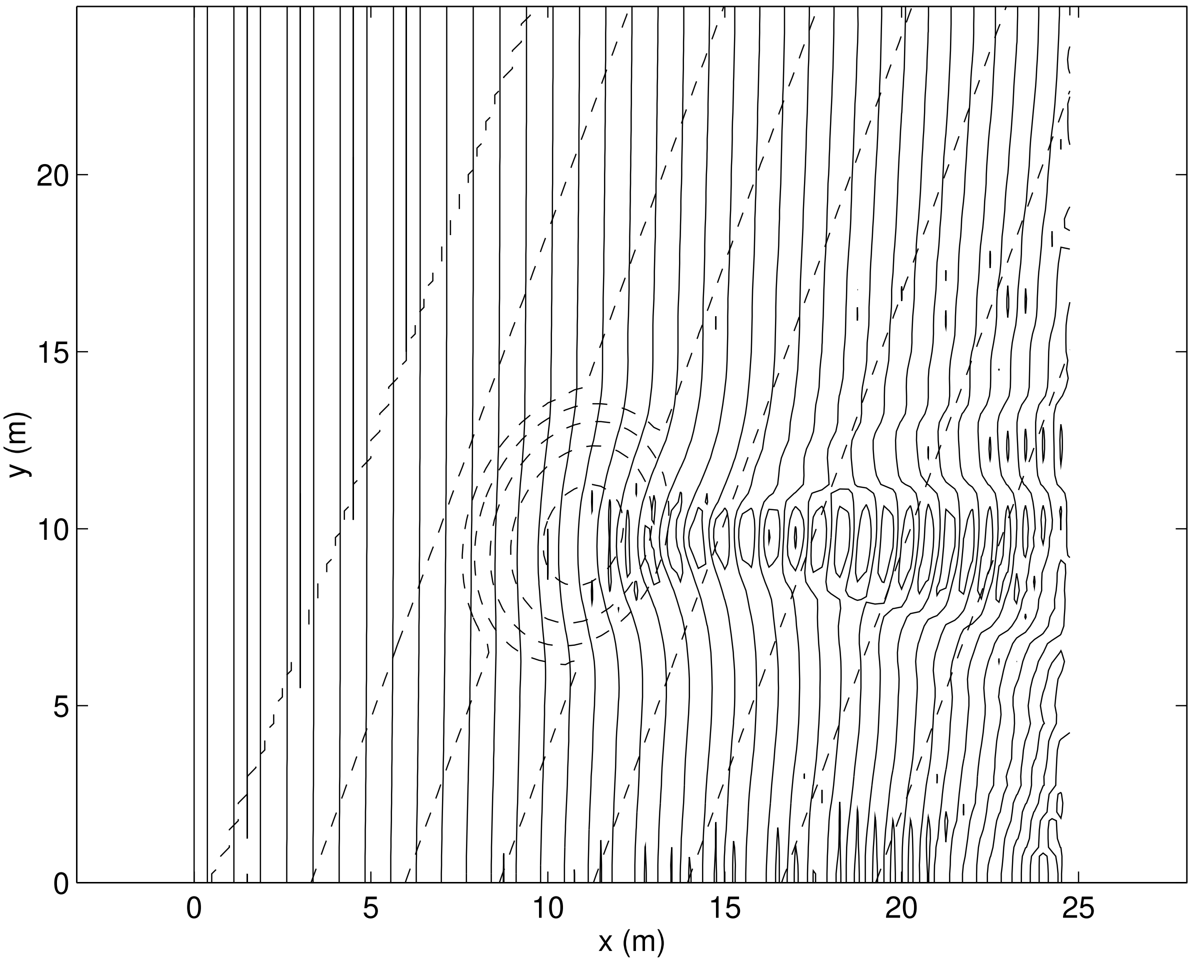

Figure 16: Results for waves propagating over a submerged shoal: surface elevation contours.

Comparisons of the predictions of this model using both the Stokes wave nonlinearity and the composite model of Kirby and Dalrymple (1986b) are given in Kirby and Dalrymple (1986b); a comparison to laboratory data is also included and shows that the model is quite accurate in predicting the wave field.

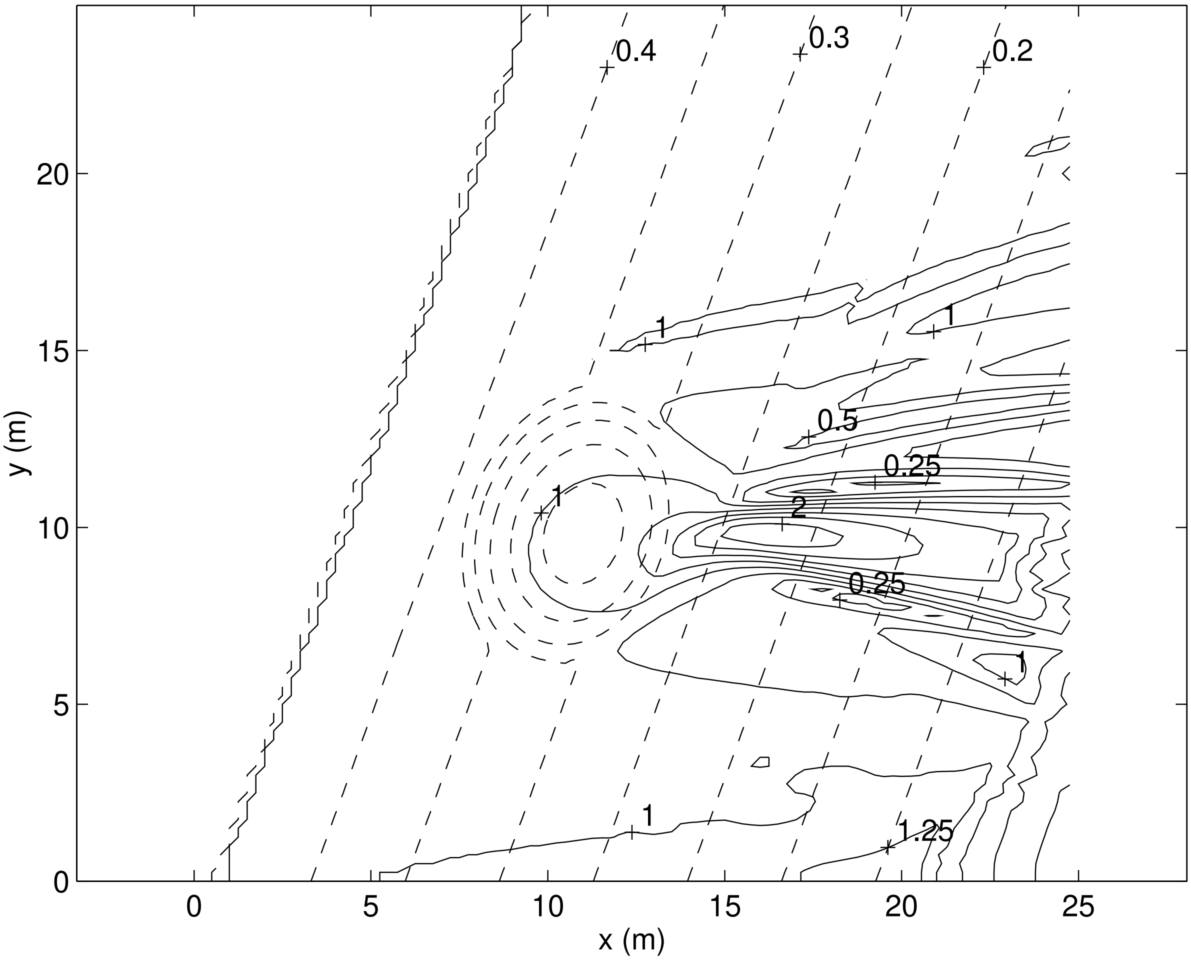

Figure 17: Results for waves propagating over a submerged shoal: wave height contours.

3.3 Waves Interacting with a Rip-Current

An example of waves which are normally incident on a planar beach and interact with a steady rip current flowing offshore from the beach has been included here in order to show the effects of wave-current interaction in the model. This example was first used by Arthur (1950) to illustrate the effects of currents and depth changes acting together on results of ray tracing schemes. However, it also provides an important example of the usefulness of the combined refraction-diffraction scheme, since it represents a case where ray tracing breaks down due to the crossing of wave rays.

The velocity field studied by Arthur is shown together with computed wave orthogonals in Figure 18. Denoting a coordinate x' pointed offshore (the opposite of the x we will be using), the velocity distribution is given by

$$U = 0.02295x'\mathrm{e}^{-(x'/76.2)^2/2}\mathrm{e}^{-(y/7.62)^2/2} \tag{34}$$

$$V = -0.2188[2-(x'/76.2)^2]\mathrm{e}^{-(x'/76.2)^2/2}\mathrm{erf}(y/76.22)\mathrm{sign}(y) \tag{35}$$

where the velocities are in m/sec. In terms of x', the bottom topography is given by

$$h(x') = 0.02x' \tag{36}$$

Arthur ran his calculations for a wave period of $T = 8$ seconds.

A photograph of the wave field created in a laboratory study of waves interacting with an ebb tidal jet is shown in Figure 19. This photograph, taken from a paper by Hales and Herbich (1972), is provided for guidance in interpreting the contour plot of surface elevations provided below.

3.3.1 Setting Up the Model

We choose a grid spacing of $dxr = 5$m and $dyr = 5$m. We choose $mr = 100$ and $nr = 100$, giving an offshore and longshore extent of 495m. The most-shoreward grid row is established 5m from shore, giving a depth range of 10m to 0.1m. Arthur's wave period is retained, and we use an initial wave amplitude of 0.1m. The input conditions are a single, normally incident wave, no user specified grid subdivision, and one frequency component.

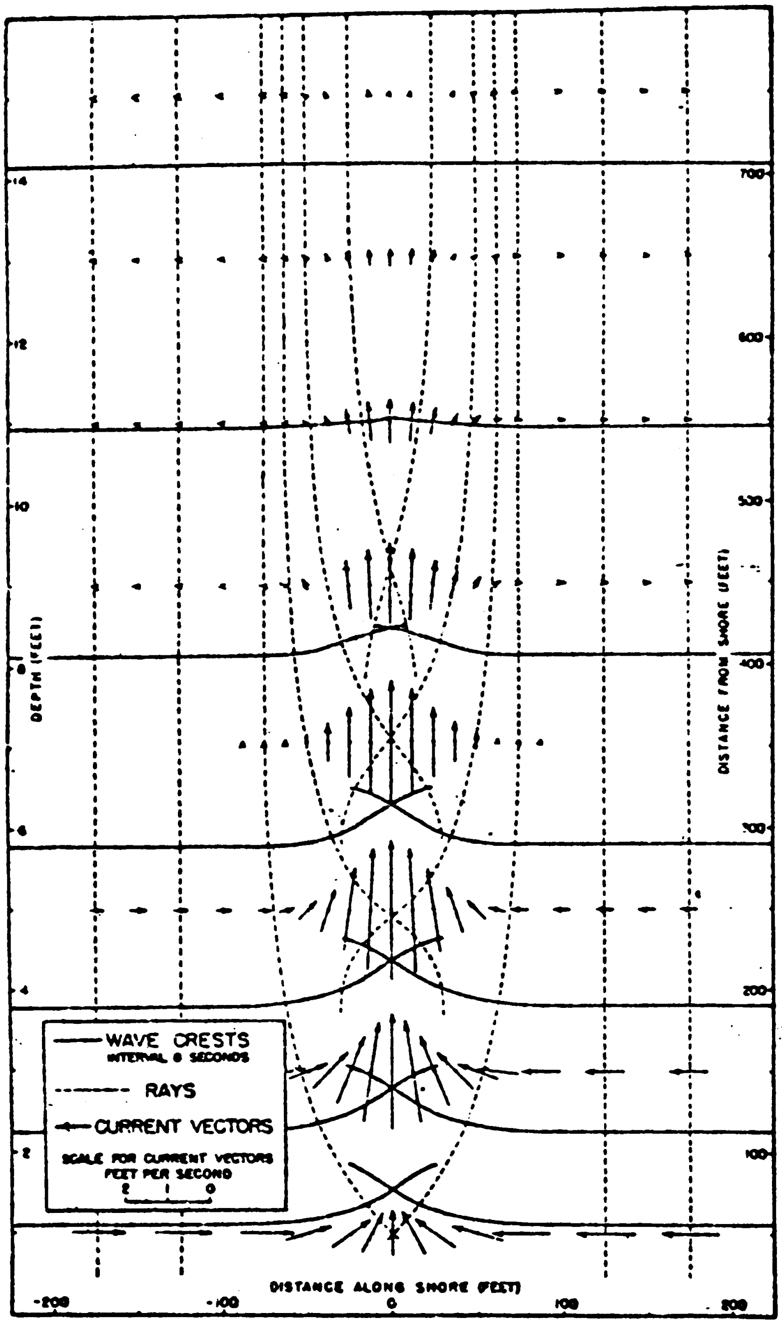

Figure 18: Pattern of orthogonals and wave crests for waves in presence of rip currents: refraction approximation. (from Arthur, 1950)

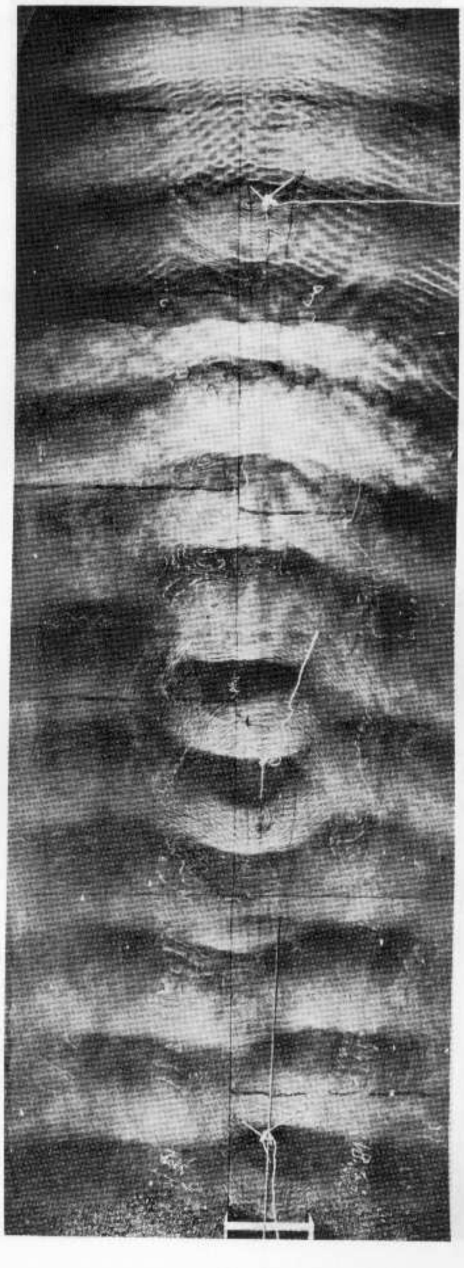

FIG. 11-WAVE TRAIN PROPAGATING ONTO AN EBB CURRENT. $U/C_o = 0.097$, $L_o/d = 5.37$, $H_o/L_o = 0.021$

Figure 19: Wave pattern on an ebb-tidal jet. (from Hales and Herbich, 1972)

The input data file for the present case follows.

```
$fnames
```

⋯ same as previous example ⋯

```
$end
$ingrid
mr =          100
nr =          100
iu =            1
ntype =           1
icur =           1
ibc =           0
dxr =  5.000000
dyr =  5.000000
dt =  10.00000
ispace =           0
nd =            1
iff =           0,           0,           0
isp =           0
iinput =           1
ioutput =           1
$end
$waves1a
iwave =           1
nfreqs =           1
$end
$waves1b
freqs =  8.000000
tide =  0.0000000E+00
nwavs =           1
amp =  0.5000000
dir =  0.0000000E+00
$end
```

The input data files may be constructed for this test case using the program *datgenv25.f*

3.3.2 Model Results

Results for this case are limited to plots of surface contours and wave height contours. These plots were constructed using the information stored in the data files *depth.dat, height.dat, surface.dat*. The plots are given in Figures 20 and 21. Note that the plots only cover the region $51 \leq ir \leq 100, 26 \leq jr \leq 75$, in order to show greater detail in the wave pattern over the rip current. The plots show a shoaling, plane wave which approaches the beach with no distortion until the wave begins to interact with the rip current. The rip causes a focussing of waves and the formation of discontinuities in the wave crests, as in the photograph in Figure 19.

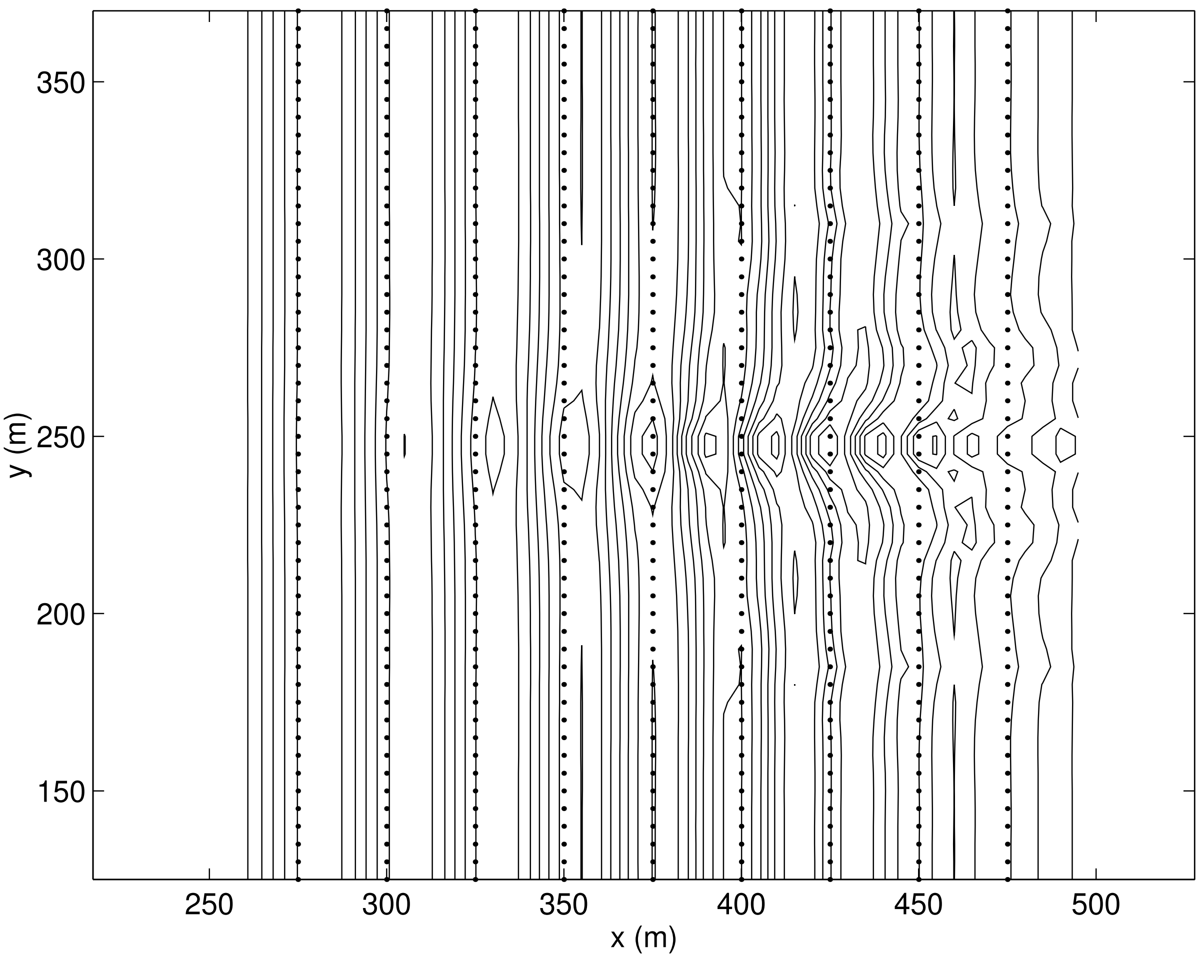

Figure 20: Waves interacting with a rip current. Shoreline at right. Surface displacement contours.

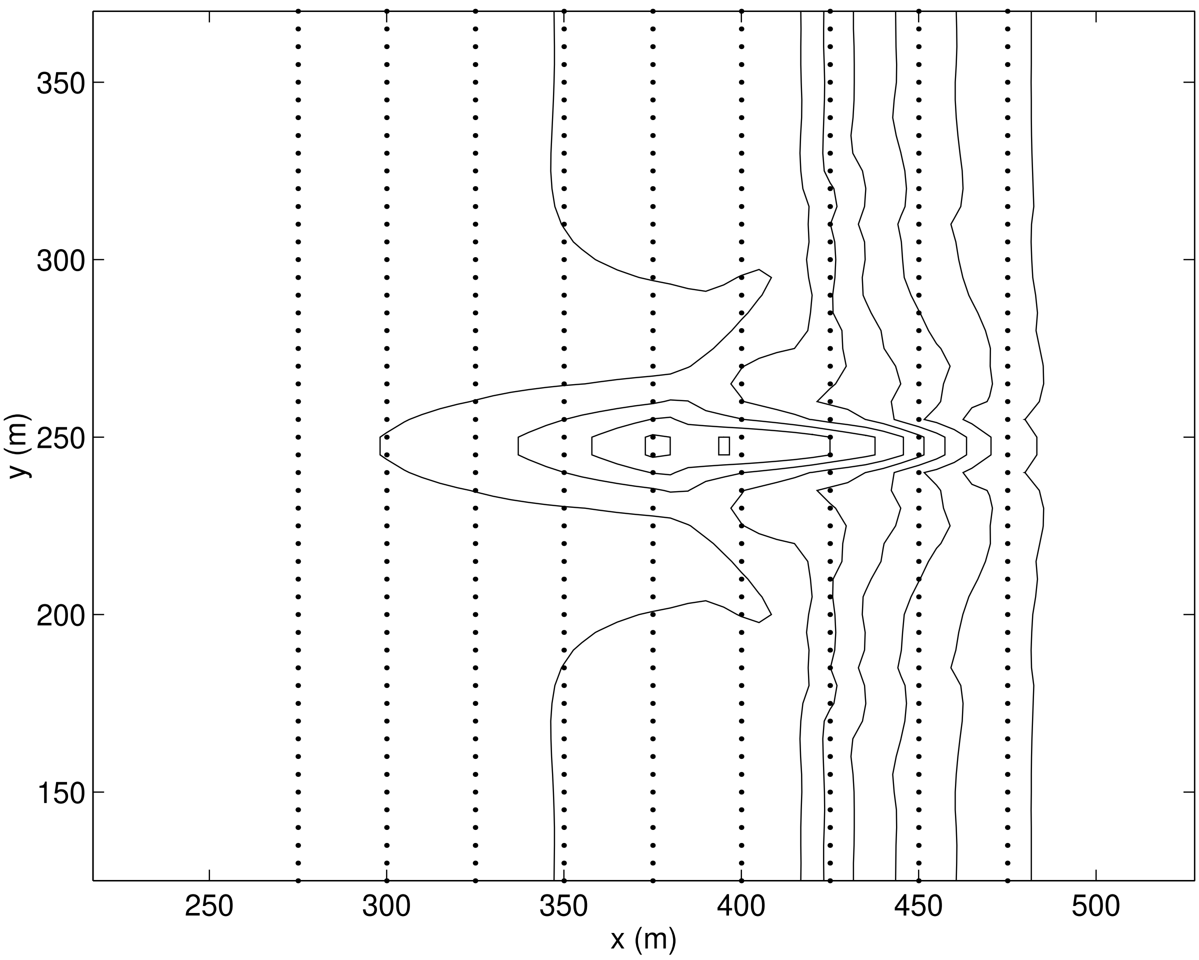

Figure 21: Waves interacting with a rip current. Shoreline at right. Wave height contours.

4 REFERENCES

Arthur, R. S., 1950, "Refraction of shallow water waves: the combined effect of currents and underwater topography", *Trans. AGU*, **31**, 549-552.

Berkhoff, J.C.W., 1972, "Computation of combined refraction-diffraction," *Proc. 13th Int. Conf. Coastal Engrg.*, ASCE, Vancouver.

Berkhoff, J.C.W., N. Booij and A.C. Radder, 1982, "Verification of numerical wave propagation models for simple harmonic linear waves," *Coastal Engineering*, 6, 255-279.

Bettess, P. and O.C. Zienkiewicz, 1977, "Diffraction and refraction of surface waves using finite and infinite elements," *Int. J. for Numerical Methods in Engrg.*, 1, 1271-1290.

Booij, N., 1981, *Gravity Waves on Water with Non-uniform Depth and Current*, Doctoral dissertation, Technical University of Delft, The Netherlands, 131 pp.

Booij, N., 1983, "A note on the accuracy of the mild-slope equation," *Coastal Engineering*, 7, 191-203.

Carnahan, B., H.A. Luther and J.O. Wilkes, 1969, *Applied Numerical Methods*, Wiley.

Chu, V.C. and C.C. Mei, 1970, "On slowly varying Stokes waves," *J. Fluid Mech.*, 41, 873-887.

Dally, W.R., R.G. Dean and R.A. Dalrymple, 1985, "Wave height variations across beaches of arbitrary profile," *J. Geophys. Research*, 90, 11917-11927.

Dalrymple, R.A., J.T. Kirby and P.A. Hwang, 1984a, "Wave diffraction due to areas of energy dissipation," *J. Waterway, Port, Coastal and Ocean Div.*, ASCE, 110, 67-79.

Dalrymple, R.A., J.T. Kirby and D.W. Mann, 1984b, "Wave propagation in the vicinity of islands," *Proc. of the 16^{th} Offshore Tech. Conf.*, No. 4675, Houston, May.

Dalrymple, R. A., 1988, "A model for the refraction of water waves", *J. Waterway, Port, Coastal and Ocean Engineering*, 114, 423-435.

Dalrymple, R. A., 1991, "REFRACT: A refraction program for water waves. Version 2.0", Report CACR-91-09, Center for Applied Coastal Research, Dept. of Civil Engrng., Univ. of Delaware, Newark.

Dean, R.G. and R.A. Dalrymple, 1984, *Water Wave Mechanics for Engineers and Scientists*, Englewood Cliffs: Prentice-Hall.

Djordjevic, V.D. and L.G. Redekopp, 1978, "On the development of packets of surface gravity waves moving over and uneven bottom," *Z. Angew. Math. and Phys.*, 29, 950-962.

Hales, L. Z. and Herbich, J. B., 1972, "Tidal inlet current-ocean wave interaction", *Proc. 13th ICCE*, 669-688.

Hedges, T.S., 1976, "An empirical modification to linear wave theory," *Proc. Inst. Civ. Eng.*, 61, 575-579.

Houston, J.R., 1981, "Combined refraction-diffraction of short waves using the finite element method", *Applied Ocean Res.*, 3, 163-170.

Jonsson, I.G. and O. Skovgaard, 1979, "A mild-slope wave equation and its application to tsunami calculations," *Mar. Geodesy*, 2, 41-58.

Kirby, J .T., 1983, "Propagation of weakly-nonlinear surface water waves in regions with varying depth and current", ONR Tech. Rept. 14, Res. Rept. CE-83-37, Department of Civil Engineering, University of Delaware, Newark.

Kirby, J.T. and R.A. Dalrymple, 1983a, "A parabolic equation for the combined refraction-diffraction of Stokes waves by mildly varying topography," *J. Fluid Mech.*, 136, 543-566.

Kirby, J.T. and R.A. Dalrymple, 1983b, "The propagation of weakly nonlinear waves in the presence of varying depth and currents," *Proc. XX^{th} Congress I.A.H.R.*, Moscow.

Kirby, J.T., 1984, "A note on linear surface wave-current interaction," *J. Geophys. Res.*, **89**, 745-747.

Kirby, J.T. and R.A. Dalrymple, 1984, "Verification of a parabolic equation for propagation of weakly non-linear waves," *Coastal Engineering*, **8**, 219-232.

Kirby, J. T., 1986a, "Higher-order approximations in the parabolic equation method for water waves", *J. Geophys. Res.*, **91**, 933-952.

Kirby, J. T., 1986b, "Rational approximations in the parabolic equation method for water waves", *Coastal Engineering*, **10**, 355-378.

Kirby, J. T., 1986c, "Open boundary condition in parabolic equation method", *J. Waterway, Port, Coast. and Ocean Eng.*, **112**, 460-465.

Kirby, J. T. and Dalrymple, R. A., 1986a, "Modeling waves in surfzones and around islands", *J. Waterway, Port, Coast. and Ocean Eng.*, **112**, 78-93.

Kirby, J. T. and Dalrymple, R. A., 1986b, "An approximate model for nonlinear dispersion in monochromatic wave propagation models", *Coast. Eng.*, **9**, 545-561.

Kirby, J. T., 1993, "Damping of high frequency noise in the large angle parabolic equation method", manuscript.

Krommes, J. A., 1992, "The *web* system of structured software design and documentation for C, C++, Fortran, Ratfor, and TEX", draft report.

Liu, P.L.-F. and R.A. Dalrymple, 1984, "The damping of gravity water waves due to percolation," *Coastal Engineering.*

Liu, P.L.-F. and T.-K. Tsay, 1984, "On weak reflection of water waves," *J. Fluid Mech.*, 131, 59-71.

MacCamy, R.D. and R.A. Fuchs, 1954, "Wave forces on piles: a diffraction theory," Tech. Memo, 69, Beach Erosion Board.

Medina, R., 1991, personal communication.

Mei, C.C. and E.O. Tuck, 1980, "Forward scattering by thin bodies," *SIAM J. Appl. Math.*, 39, 178-191.

Mitsuyasu, H., F. Tasai, T. Suhara, S. Mizuno, M. Ohkusu, T. Honda and K. Rikishii, 1975, "Observations of the directional spectrum of ocean waves using a cloverleaf buoy," *J. Physical Oceanography*, 5, 750-760.

Phillips, O.M., 1966, *The Dynamics of the Upper Ocean*, Cambridge University Press.

Radder, A.C., 1979, "On the parabolic equation method for water-wave propagation," *J. Fluid Mech.*, 95, 159-176.

Sommerfeld, A., 1896, "Mathematische theorie der diffraction", *Math. Annalen*, 47, 317-374.

U.S. Army Coastal Engineering Research Center, 1973, *Shore Protection Manual*, Vol. I.

Wiegel, R.L., 1962, "Diffraction of waves by a semi-infinite breakwater," *J. Hydraulic Div.*, ASCE, 88, HY1, 27-44.

Yue, D.K.P. and C.C. Mei, 1980, "Forward diffraction of Stokes waves by a thin wedge," *J. Fluid Mech.*, 99, 33-52.

5 Appendix A: *fweb* Documentation of the REF/DIF 1 Program Listing

The following section contains a document which provides a heavily anotated program listing for **REF/DIF 1**, which has been produced using the *fweb* documentation program (Krommes, 1992). The use of *fweb* here is somewhat experimental. In theory, *fweb* provides a programming environment which allows the programmer to specify the operation of a block of code in full, typeset detail, after which the actual *Fortran* or *C* code is spelled out. This procedure places a high premium on the use of a highly structured and modularized programming technique. In practice, many longer modelling codes are being essentially "retrofitted" for *fweb*, and thus the documentation does not go much beyond upgrading the comment statements appearing in standard *Fortran* style. As **REF/DIF 1** evolves, the *fweb* documentation will become a more interwoven portion of the documentation and user's manual.

1. Refraction-Diffraction Model REF/DIF 1, Version 2.5.

REF/DIF 1 calculates the forward scattered wave field in regions with slowly varying depth and current, including the effects of refraction and diffraction. The program is based on the parabolic equation method. Physical effects included in the present version include:

1. Parabolic approximation:
   (a) Minimax approximation given by Kirby (1986b).
2. Wave nonlinearity: choice of
   (a) Linear.
   (b) Composite nonlinear: approximate model of Kirby and Dalrymple (1986b).
   (c) Stokes nonlinear: model of Kirby and Dalrymple (1983a).
3. Wave breaking:
   (a) Model of Dally, Dean and Dalrymple(1985).
4. Absorbing structures and shorelines:
   (a) Thin film model surrounded by a natural surfzone (Kirby and Dalrymple, 1986a).
5. Energy dissipation: any of
   (a) Turbulent bottom friction damping.
   (b) Porous bottom damping.
   (c) Laminar boundary layer damping.
6. Lateral boundary conditions: either of
   (a) Reflective condition.
   (b) Open boundary condition (Kirby, 1986c).
7. Input wave field: either of
   (a) Model specification of monochromatic or directional wave field.
   (b) Input of initial row of data from disk file.
8. Output wave field:
   (a) Standard output.
   (b) Optional storage of last full calculated row of complex amplitudes.

The documentation of present program is contained in:

Combined Refraction/Diffraction Model REF/DIF 1, Version 2.5, Documentation and User's Manual
James T. Kirby and Robert A. Dalrymple
CACR Report No. 94 - 04 , Center for Applied Coastal Research
Department of Civil Engineering, University of Delaware, July 1994.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Program developed by James T. Kirby and Robert A. Dalrymple.
Version 2.5, Last revision 12/21/94.

program *refdif1*

include `'param.h'`

⟨ common statements 18 ⟩

// Constants which provide for conversion between MKS and English units on input and output.

$dconv(1) = 1.$
$dconv(2) = 0.30488$

// read control parameters and reference grid data

call *inref*

// read control parameters and initializing wave data

call *inwave*

// Pass program control to subroutine *model*.

// For each frequency component specified in *inwave*, *model* executes the model throughout the entire grid and then reinitializes the model for the next frequency.

call *model*

// All done. Close output data files if **open** and **close** statements are being used.

```
   // outdat.
  close (iun(3))
   // surface.
  if (fname6 ≠ ' ') close (8)
   // angle.
  close (9)
   // refdif1.log.
  close (10)
   // height.
  close (12)
   // sxx.
  if (fname12 ≠ '␣') close (13)
   // sxy.
  if (fname13 ≠ '␣') close (14)
   // syy.
  if (fname14 ≠ '␣') close (15)
   // depth.
  close (16)
   // bottomu.
  if (fname7 ≠ '␣') close (17)
   // owave.
  if (ioutput ≡ 2) close (33)
  stop
end
```

2. INREF.

Subroutine reads in and checks dimensions and values for large scale reference grid. Wave parameters for the particular run are read in later by subroutine *inwave*.

The following unit (device) numbers are assumed:

- $iun(1)$: input reference grid values of d, u, and v
- $iun(2)$: Input user specified subgrid divisions from file normally named *subdat.dat*.
- $iun(3)$: Output results at reference grid locations to disk file.
- 8: Output image of instantaneous water surface at computational grid resolution. This is interpolated to a regular rectangular grid by the program *surface2hdf.f* and stored in HDF file format. Usual name for file is *surface.dat*.
- 9: Output results for wave angles in file usually named *angle.dat*.
- 10: Log file *refdif1*.log, containing basic information and error messages.
- 11: Input file *wave.dat*, user specifies complex amplitude on first row.
- 12: Output results for significant wave height in file usually named *height.dat*.
- 13: Output results for rad. stress Sxx in file usually named *sxx.dat*.
- 14: Output results for rad. stress Sxy in file usually named *sxy.dat*.
- 15: Output results for rad. stress Syy in file usually named *syy.dat*.
- 16: Output results for tide-corrected depth grid in *depth.dat*
- 17: Output results for tide-corrected water depths in file usually named *depth.dat*.
- $iun(5)$: Unit for file containing *namelist* input data. Usually named *indat.dat*. This filename is specified in the standard program version. In the LRSS version, an arbitrary filename is entered on the command line.

Variable definitions.

mr, nr	reference grid dimensions (max ixr, iyr)
dxr, dyr	grid spacing for reference grid
iu	physical unit descriptor (1=mks, 2=english) default value is 1, mks units
dt	depth tolerence value (to check for anomalous depth values)
$ispace$	switch to control grid subdivision. =0, program attempts its own subdivisions =1, user specifies subdivisions
nd	y direction subdivision ($ispace = 0$ or 1) (must be $.lt.iy/nr - 1$)
$md(mr-1)$	x direction subdivisions (if $ispace = 1$) (must be $.le.ix - 1$)
$ntype$	nonlinearity control parameter =0, linear model =1, Stokes matched to Hedges in shallow water =2, Stokes throughout
$icur$	switch to tell program if current data is to be used and read on input =0, no input current data =1, input current data to be read program defaults to $icur = 0$
ibc	boundary condition switch =0, use closed lateral boundaries =1, use open lateral conditions program defaults to $ibc = 0$
dr	depths at reference grid points > 0, submerged areas < 0, elevation above surface datum
ur	x velocities at reference grid points (only entered if $icur = 1$)
vr	y velocities at reference grid points (only entered if $icur = 1$)

Data is entered in *namelist* format from the data file *indat.dat*.

Subroutine is called from *refdif1* and returns control to calling location, unless a fatal error is encountered during input data checking.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by James T. Kirby, October 1984. Revised July 1994.

subroutine *inref*

include `'param.h'`

⟨ common statements 18 ⟩

// Standard file name choices:

// $fname1$ = $refdat.dat$, reference grid data file.

// $fname2$ = $outdat.dat$, standard output data file.

// $fname3$ = $subdat.dat$, user-specified subgrids.

// $fname4$ = $wave.dat$, user-specified complex amplitude on row 1 (for $iinput$ =2).

// $fname5$ = $owave.dat$, complex amplitude on last row (for $ioutput$ = 2).

// $fname6$ = $surface.dat$, instantaneous water surface at computational resolution.

// $fname7$ = $bottomu.dat$, magnitude of bottom velocity at reference grid points.

// $fname8$ = $angle.dat$, wave directions at reference grid points.

// $fname9$ = not used yet.

// $fname10$ = $refdif1$.log, run log for $refdif1$ program.

// $fname11$ = $height.dat$, wave heights at reference grid locations.

// $fname12$ = $sxx.dat$, Sxx components at reference grid locations.

// $fname13$ = $sxy.dat$, Sxy components at reference grid locations.

// $fname14$ = $syy.dat$, Syy components at reference grid locations.

// $fname15$ = $depth.dat$, tide-corrected depths at reference grid locations.

// $fnamein$ = $indat.dat$, input namelist file.

$namelist/ingrid/mr, nr, iu, ntype, icur, ibc, dxr, dyr, dt, ispace, nd, iff, isp, iinput, ioutput/inmd/md/fnames/fname1, fname2, fname3, fname4, fname5, fname6, fname7, fname8, fname9, fname10, fname11, fname12, fname13, fname14, fname15$

// Constants.

$g = 9.80621$

3. Specify name of namelist data file.

The LRSS implementation of Ref/Dif 1 imposes the restriction that no file names can be specified within the program itself. This makes it necessary to read in at least one file name as a command line argument. Two options are provided here by means of a subroutine *infile*. The code for the subroutine is provided in either of the files

1. *infile1.f* - standard version. The program assumes the name *indat.dat*.

2. *infile2.f* - user specifies the file name using the *igetarg* command line syntax.

The user of *refdif1* must copy the desired file to *infile.f* before compiling the program. The *igetarg* structure is supported on Sun Fortran and may be used at all times there. The SGI version tested to date uses a subroutine library *liblrss.a* provided by SAIC.

call *infile*(*fnamein*)

$iun(5) = 5$

open ($unit = iun(5), file = fnamein, status =$ `'old'`)

4. **Read remaining file names from namelist.**

$iun(1) = 1$
$iun(2) = 2$
$iun(3) = 3$

read $(iun(5), nml = fnames)$

open $(unit = iun(1), file = fname1, status =$ 'old'$)$
open $(unit = iun(3), file = fname2)$
open $(9, file = fname8)$
open $(10, file = fname10)$
open $(12, file = fname11)$

if $(fname12 \neq$ '␣' $\wedge fname13 \neq$ '␣' $\wedge fname14 \neq$ '␣'$)$ **then**
  open $(13, file = fname12)$
  open $(14, file = fname13)$
  open $(15, file = fname14)$
endif

open $(16, file = fname15)$

if $(fname7 \neq$ '␣'$)$ **open** $(17, file = fname7)$

if $(fname6 \neq$ '␣'$)$ **open** $(8, file = fname6)$

  // print headers on output

write $(10, 120)$

write $(10, 106)$

  // Read control data from unit $iun(5)$.

read $(iun(5), nml = ingrid)$

if $(ispace \equiv 1)$ **read** $(iun(5), nml = inmd)$

write $(10, 107)$ mr, nr, dxr, dyr

if $(iu \equiv 1)$ **write** $(10, 114)$ iu
if $(iu \equiv 2)$ **write** $(10, 115)$ iu

if $(icur \equiv 0)$ **write** $(10, 200)$
if $(icur \equiv 1)$ **write** $(10, 201)$

if $(ibc \equiv 0)$ **write** $(10, 202)$

```
if (ibc ≡ 1) write (10, 203)

if (ispace ≡ 0) write (10, 108)
if (ispace ≡ 1) write (10, 109)

write (10, 119) nd

if (ntype ≡ 0) write (10, 110)
if (ntype ≡ 1) write (10, 111)
if (ntype ≡ 2) write (10, 112)

  // Check input from unit iun(5).

if ((mr > ixr) | (nr > iyr)) then
  write (10, *) 'dimensions␣for␣reference␣grid␣too␣large,␣stopping'
  call exit(1)
end if

if ((iu ≠ 1) ∧ (iu ≠ 2)) iu = 1

dt = dt*dconv(iu)
dxr = dxr*dconv(iu)
dyr = dyr*dconv(iu)

if (dt ≡ 0.) dt = 2.

if (nd > ifix(float(iy − 1)/float(nr − 1))) then
  write (10, 102)
  call exit(1)
endif

if (ispace ≡ 1) then
  test = 0.
  do 1 i = 1, mr − 1
    if (md(i) > (ix − 1)) then
      write (10, 103) i
      test = 1.
    endif
1: continue
  if (test ≡ 1.) call exit(1)
endif

  // read depth grid and velocities from unit iun(1)

do 2 i = 1, mr
```

```
    read (iun(1), 101) (dr(i,j), j = 1, nr)
2: continue
  if (icur ≡ 1) then
    do 3 i = 1, mr
      read (iun(1), 101) (ur(i,j), j = 1, nr)
  3: continue
    do 4 i = 1, mr
      read (iun(1), 101) (vr(i,j), j = 1, nr)
  4: continue
  endif
    // convert depth and currents
  do 5 i = 1, mr
    do 5 j = 1, nr
      dr(i,j) = dr(i,j)*dconv(iu)
5: continue
  if (icur ≡ 1) then
    do 55 i = 1, mr
      do 55 j = 1, nr
        ur(i,j) = ur(i,j)*dconv(iu)
        vr(i,j) = vr(i,j)*dconv(iu)
 55: continue
  endif
    // check for large depth changes and large currents in reference grid data.
  do 6 i = 2, mr - 1
    do 6 j = 2, nr - 1
      dcheck = (dr(i+1,j) + dr(i-1,j) + dr(i,j-1) + dr(i,j+1))/4.
      if (abs(dcheck - dr(i,j)) > dt) write (10, 104) dr(i,j), i, j, dt
6: continue
  if (icur ≡ 1) then
    do 7 i = 1, mr
      do 7 j = 1, nr
        if (dr(i,j) ≤ 0.0) go to 7
        fr = (ur(i,j)*ur(i,j) + vr(i,j)*vr(i,j))/(g*dr(i,j))
```

if $(fr > 1.)$ **write** $(10, 105)$ $i,\ j,\ fr$

7: **continue**

endif

// Establish coordinates for reference grid.

do 8 $ir = 1, mr$

$xr(ir) = \mathsf{float}(ir - 1) * dxr$

8: **continue**

do 9 $jr = 1, nr$

$yr(jr) = \mathsf{float}(jr - 1) * dyr$

9: **continue**

// Establish y coordinates for interpolated grid.

$n = nd * (nr - 1) + 1$

$dy = dyr / \mathsf{float}(nd)$

do 10 $j = 1, n$

$y(j) = \mathsf{float}(j - 1) * dy$

10: **continue**

// Write grid information on output unit $iun(3)$.

write $(iun(3), *)$ $nr,\ mr$

write $(iun(3), *)$ $(yr(jr)/dconv(iu), jr = 1, nr)$

// Check friction values.

// $iff(1) = 1$, turbulent boundary layer damping everywhere

// $iff(2) = 1$, porous bottom damping everywhere

// $iff(3) = 1$, laminar boundary layer damping everywhere

do 11 $i = 1, 3$

if $((iff(i) \neq 0) \wedge (iff(i) \neq 1))$ $iff(i) = 0$

11: **continue**

write $(10, 116)$ $(iff(i), i = 1, 3)$

// Specify whether or not user specified subgrids are to be read in during model operation.

// $isp = 0$, no subgrids specified

// $isp = 1$, subgrids to be read in later from unit $iun(2)$

if $(isp \equiv 0)$ **write** $(10, 117)$

```
if (isp ≡ 1) then
   write (10, 118)
   open (unit = iun(2), file = fname3, status = 'old')
endif

if ((isp ≡ 1) ∧ (ispace ≡ 0))  write (10, 113)

if (isp ≡ 0) then

   do 14 ir = 1, mr
      do 14 jr = 1, nr
         isd(ir, jr) = 0
14: continue

else

   do 15 ir = 1, mr − 1
      read (iun(2), 100) (isd(ir, jr), jr = 1, nr − 1)
15: continue

endif

   // Input done, return to main program.

return

100: format (15i4) ;
101: format (20f10.4) ;
102: format (' y-direction subdivision too fine.'/' maximum number of y grid\
        points will be exceeded.'/' execution terminating.') ;
103: format (' x-direction subdivision too fine on grid block', 2x,
        i3/' execution terminating') ;
104: format (' depth', 2x,
        f7.2, '(m) at reference grid location', 2(2x, i3)/' differs from the average\
        of its neighbors by', ' more than', 2x, f7.2, '(m).'/' execution continuing')
        ;
105: format (' ambient current at reference grid location', 2(2x, i3),
        ' is supercritical with froude number =', f7.4/' execution continuing') ;
106: format ('0'///20x, 'input section, reference grid values'///) ;
107: format (' reference grid dimensions  mr=', i3/'                          \
           nr=', i3///' reference grid spacings   dxr=',
        f8.4/'                            dyr=', f8.4) ;
```

```
108: format('␣'/'␣ispace␣=0␣chosen,␣program␣will␣attempt␣its␣own␣',
        'reference␣grid␣subdivisions');
109: format('␣'/'␣ispace␣=1␣chosen,␣subdivision␣spacings␣will␣be','␣input␣as␣data')
        ;
110: format('␣'/'␣ntype␣=␣0,␣linear␣model');
111: format('␣'/'␣ntype␣=␣1,␣stokes␣model␣matched␣to␣hedges␣model');
112: format('␣'/'␣ntype␣=␣2,␣stokes␣model');
113: format('␣warning:␣input␣specifies␣that␣user␣will␣be␣supplying',
        '␣specified␣subgrids␣(isp=1),'/'␣while␣program␣has␣been␣t\
        old␣to␣generate␣its␣own␣subgrid',
        '␣spacings␣(ispace=0).'/'␣possible␣incompatibility\
        ␣in␣any␣or␣all␣subgrid␣blocks');
114: format('␣'/'␣physical␣unit␣switch␣iu=',i1,',␣␣input␣in␣mks␣units');
115: format('␣'/'␣physical␣unit␣switch␣iu=',i1,',␣␣input␣in␣english␣units');
116: format('␣'//'␣␣␣switches␣for␣dissipation␣terms'//'␣',i1,
        '␣␣␣turbulent␣boundary␣layer'/'␣',i1,'␣␣␣porous␣bottom'/'␣',i1,
        '␣␣␣laminar␣boundary␣layer');
120: format(//////20x,
        'Refraction-Diffraction␣Model␣for'/20x,'Weakly␣Nonlinear␣Surface␣\
        Water␣Waves'///20x,'REF/DIF␣1,␣␣Version␣2.5'///20x,
        'Center␣for␣Applied␣Coastal␣Research'/20x,'Department␣of␣Civil␣Engin\
        eering'/20x,'University␣of␣Delaware'/20x,'Newark,␣Delaware␣19716'///10x,
        'James␣T.␣Kirby␣and␣Robert␣A.␣Dalrymple,␣November␣1994');
117: format('␣'/'␣isp=0,␣no␣user␣defined␣subgrids');
118: format('␣'/'␣isp=1,␣user␣defined␣subgrids␣to␣be␣read');
119: format('␣'/'␣y-direction␣subdivision␣according␣to␣nd=',i3);
200: format('␣'/'␣icur=0,␣no␣current␣values␣read␣from␣input␣files');
201: format('␣'/'␣icur=1,␣current␣values␣read␣from␣data␣files');
202: format('␣'/'␣ibc=0,␣closed␣(reflective)␣lateral␣boundaries');
203: format('␣'/'␣ibc=1,␣open␣lateral␣boundaries');
     end
```

5. INWAVE.

Read in wave parameters.

Variable definitions:

iinput	determine method of specifying the first row of computational values
	=1, input values from indat.dat according to value of *iwave*
	=2, input complex a values from file *wave.dat*
ioutput	determine whether last row of complex amplitudes are stored in separate file *owave.dat*
	=1, extra data not stored
	=2, extra data stored in file *owave.dat*

if $iinput = 1$:

iwave	input wave type
	=1, input discrete wave amplitudes and directions
	=2, read in dominant direction, total average energy density, and spreading factor
nfreqs	number of frequency components to be used (separate model run for each component)
freqs	wave frequency for each of *nfreqs* runs
tide	tidal offset for each of *nfreqs* runs

if $iwave = 1$

nwavs	number of discrete wave components at each of *nfreqs* runs
amp	initial amplitude of each component
dir	direction of each discrete component in + or - degrees from the x-direction

if $iwave = 2$

edens	total amplitude variance density over all directions at each frequency
$nsp = n$	spreading factor in $\cos^{2n}(\theta)$ directional distribution (stored in *nwavs*)
nseed	Seed value for random number generator. Integer value between 0 and 9999.

if $iinput = 2$:

freqs	Wave frequency for one run.
tide	Tidal offset for one run.

All data is entered using the *namelist* convention.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by James T. Kirby, Oct 1984, Sept 1989, Jan 1991, July 1994, November 1994.

subroutine *inwave*

include `'param.h'`

⟨ common statements 18 ⟩

```
namelist/waves1a/iwave, nfreqs/waves1b/freqs, tide, nwavs, amp,
    dir/waves1c/thet0, freqs, tide, edens, nwavs, nseed/waves2/freqin, tidein
pi = 3.1415927
  // Values of iinput, ioutput already entered in namelist statement in inref.
if ((iinput ≠ 1) ∧ (iinput ≠ 2)) then
  write (10, *) '␣invalid␣value␣chosen␣for␣iinput,␣check␣indat.dat'
  stop
endif
if ((ioutput ≠ 1) ∧ (ioutput ≠ 2)) then
  write (10, *) '␣invalid␣value␣chosen␣for␣ioutput,␣check␣indat.dat'
  stop
endif
if (ioutput ≡ 2) then
  open (33, file = fname5)
endif
if (iinput ≡ 1) then
  write (10, *) '␣iinput␣=␣1,␣program␣specifies␣initial␣row␣of␣a'
    // Enter iwave, nfreqs for iinput = 1.
  read (iun(5), nml = waves1a)
  write (10, 102)
    // Enter data for case of iinput = 1, iwave = 1.
  if (iwave ≡ 1) then
    read (iun(5), nml = waves1b)
    write (10, 103)
  endif
    // Enter data for case of iinput = 1, iwave = 2.
  if (iwave ≡ 2) then
    read (iun(5), nml = waves1c)
    write (10, 104)
  endif
  write (10, 105) nfreqs
```

if $(iwave \equiv 2)$ **then**

$thet0 = thet0 * pi/180.$

endif

// For each frequency, enter the wave period and tidal offset.

do 3 $ifreq = 1, nfreqs$

write $(10, 107)$ $ifreq$, $freqs(ifreq)$, $tide(ifreq)$

// Convert angles to radians.

$freqs(ifreq) = 2.*pi/freqs(ifreq)$
$tide(ifreq) = tide(ifreq)*dconv(iu)$

// If $iwave = 1$, read the number of discrete components.

if $(iwave \equiv 1)$ **then**

do 1 $iwavs = 1, nwavs(ifreq)$

write $(10, 106)$ $iwavs$, $amp(ifreq,iwavs)$, $dir(ifreq,iwavs)$

$dir(ifreq, iwavs) = dir(ifreq, iwavs)*pi/180.$
$amp(ifreq, iwavs) = amp(ifreq, iwavs)*dconv(iu)$

1: **continue**

endif

// If $iwave = 2$, read the parameters for each frequency.

if $(iwave \equiv 2)$ **then**

$seed = \mathsf{float}(nseed)/9999.$
write $(10, 108)$ $edens(ifreq)$, $nwavs(ifreq)$, $nseed$
$dir(ifreq, 1) = thet0$
$edens(ifreq) = edens(ifreq)*(dconv(iu)^{2.})$

endif

3: **continue**

endif

// If $iinput = 2$, read in wave period and tidal offset.

if $(iinput \equiv 2)$ **then**

read $(iun(5), nml = waves2)$

$freqs(1) = freqin$

```
      tide(1) = tidein
      write (10,*) ' iinput = 2, user specifies a in wave.dat'
      nfreqs = 1
      write (10,102)
      write (10,*) ' wave period =', freqs(1), ' sec.'
      write (10,*) ' tidal offset=', tide(1)
      freqs(1) = 2.*pi/freqs(1)
      tide(1) = tide(1)*dconv(iu)
    endif
    return
100: format(15i4) ;
101: format(20f10.4) ;
102: format('1'///20x, ' input section, wave data values'///) ;
103: format(' '///' iwave=1, discrete wave amps and directions') ;
104: format(' '///' iwave=2, directional spreading model chosen') ;
105: format(' '///' the model is to be run for', i3, ' separate',
       ' frequency components') ;
106: format(' '/' wave component ', i2, ', amplitude =', f8.4, ', direction=', f8.4) ;
107: format(' '//' frequency component ', i2//' wave period=', f8.4, 'sec.,\
        tidal offset=', f8.4) ;
108: format(' '/' total variance density =', f8.4, ', spreading factor        \
        n=', i2, ' seed number =', i5) ;
  end
```

6. MODEL.

This subroutine is the control level for the actual wave model. Data read in during *inref* and *inwave* is conditioned and passed to the wave model. This routine is executed once for each frequency component specified in *inwave*.

The wave model is split in three parts which are run sequentially for each reference grid row.

grid	subroutine performs the interpolation of depth and current values.
con	calculate the constants needed by the finite difference scheme.
fdcalc	perform the finite difference calculations.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by James T. Kirby, November 1984.

Corrections made to specification of average k value on first row, April 5, 1993.

Corrections and additions for LRSS compatibility, July-November 1994.

subroutine *model*

include `'param.h'`

⟨ common statements 18 ⟩

dimension $dthi(31)$, $thi(31)$, $thet(iy)$

// Constants.

$g = 9.80621$
$rho = 1000.$
$pi = 3.1415927$
$eps = 1.0 \cdot 10^{-05}$

// Execute model once for each frequency.

// *ifreq* is the controlling index value.

do 200 $ifreq = 1, nfreqs$
  $psibar = 0.$
  write (10, 203) *ifreq*

// Specify initial nonlinear parameters for each run.

if $(ntype \equiv 0)$ $an = 0.$

if $(ntype \neq 0)$ $an = 1.$
if $(ntype \neq 2)$ $anl = 0.$
if $(ntype \equiv 2)$ $anl = 1.$

// Calculate the mean kb on the first row, for use in specifying initial conditions.

$npts = 0$
$sumk = 0.$

do 10 $jr = 1, nr$
$d(1, jr) = dr(1, jr) + tide(ifreq)$
call $wvnum(d(1, jr), ur(1, jr), freqs(ifreq), k(1, jr), eps, icdw, 1, 1)$

if $(d(1, jr) > 0.05)$ **then**
$sumk = sumk + k(1, jr)$
$npts = npts + 1$
endif

10: **continue**

$kb(1) = sumk/\mathsf{float}(npts)$

// Establish initial wave conditions for the $ifreq$ frequency

if $(iinput \equiv 1)$ **then**

// Compute wave from data given in $indat.dat$.

if $(iwave \equiv 1)$ **then**

// $iwave \equiv 1$, discrete components specified.

do 3 $j = 1, n$
$a(1, j) = \mathsf{cmplx}(0., 0.)$
do 2 $iwavs = 1, nwavs(ifreq)$
$thet(j) = dir(ifreq, iwavs)*180./pi$
$a(1, j) = a(1, j) + amp(ifreq, iwavs)*\mathsf{cexp}(\mathsf{cmplx}(0., kb(1)*\mathsf{sin}(dir(ifreq,$
$iwavs))*y(j)))$
2: **continue**
3: **continue**
write $(9, 202)$ $(thet(j), j = 1, n, nd)$
else

// $iwave \equiv 2$, directional spreading model.

$sp = \mathsf{float}(nwavs(ifreq))$

```
nsp = nwavs(ifreq)
thmax = pi/4.
call acalc(thmax,nsp,a1)
edens(ifreq) = sqrt(edens(ifreq)/a1)
nn = 31
ii = (nn - 1)/2 + 1
seed = rand(seed)
   // Compute randomly distributed Δθ's.
sum0 = 0.
do 12 i = 1, nn
   seed = rand(seed)
   dthi(i) = seed
   sum0 = sum0 + seed
12: continue
xnorm = 2.*thmax/sum0
do 101 i = 1, nn
   dthi(i) = dthi(i)*xnorm
101: continue
thi0 = -thmax
do 4 i = 1, nn
   thi0 = thi0 + dthi(i)
   thi(i) = thi0 - dthi(i)/2.
   dth = dthi(i)
   amp(ifreq, i) = edens(ifreq)*sqrt(dth)*sqrt(cos(thi(i) + dth/2.)^(2*nsp) +
         cos(thi(i) - dth/2.)^(2*nsp))
4: continue
do 5 i = 1, nn
   ip1 = i + 1
   seed = rand(seed)
   dir(ifreq, ip1) = 2.*pi*seed/100.
5: continue
do 7 j = 1, n
   a(1, j) = cmplx(0., 0.)
   do 6 i = 1, nn
      a(1, j) = a(1, j) + amp(ifreq, i)*cexp(cmplx(0.,
            kb(1)*sin(thi(i) - thet0)*y(j) + dir(ifreq, i + 1)))*2.
```

6: **continue**

7: **continue**

endif

endif

// If $iinput = 2$, read a from data file *fname4*.

if ($iinput \equiv 2$) **then**

open (11, *file* = *fname4*)

read $(11, *)$ $(a(1,j), j = 1, n)$

close (11)

endif

// Store first row of wave heights on unit 12.

write (12, 202) $(2*\mathsf{cabs}(a(1,j))/dconv(iu), j = 1, n, nd)$

// If *fname6* not null, store surface on file *fname6*.

$x(1) = 0$

if ($fname6 \neq$ '␣') **then**

write $(8, *)$ n

write $(8, *)$ $(y(j), j = 1, n)$

write $(8, *)$ $x(1)$

write $(8, *)$ (**real** $(a(1,j))$, $j = 1$, n)

endif

// Now execute model for the *ifreq* frequency over each of *mr* grid blocks. *ir* is the controlling index value.

do 100 $ir = 1, (mr - 1)$

// Establish interpolated grid block for segment *ir*.

call *grid*(*ifreq*,*ir*)

// If $ir = 1$ write initial values on $iun(3)$.

if ($ir \equiv 1$) **then**

write (10, 201) $x(1)/dconv(iu)$, *psibar*

write $(iun(3), *)$ $x(1)/dconv(iu)$, *psibar*

write $(iun(3), *)$ $(a(1,j)/dconv(iu), j = 1, n, nd)$

write (16, 202) $(d(1,j)/dconv(iu), j = 1, n, nd)$

endif

// Calculate constants for each grid block.

call *con*(*ifreq*,*ir*)

// Perform finite difference calculations.

call *fdcalc*(*ifreq*,*ir*)

// Grid block *ir* done, print output and go to next grid.

100: **continue**

if ($ioutput \equiv 2$) **then**

write $(33, *)$ $(a(m, j), j = 1, n)$

endif

// Termination for the *surface.dat* file.

if (*fname6* $\neq$ '␣') **then**

$x(1) = -100.$

write $(8, *)$ $x(1)$

endif

// Model complete for the *ifreq* frequency component, go to the next frequency component.

200: **continue**

// Runs completed for all frequencies. Return to end of main program.

return

201: **format** ('␣x=', *f10.2*, '␣␣psibar=', *f20.4*) ;

202: **format** ('␣', 200(*f10.4*)) ;

203: **format** ('1', 20x, 'model␣execution,␣frequency', '␣component', *i4*//) ;

end

7. GRID.

Interpolate the depth and current grids for reference grid block *ir*.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by James T. Kirby, October 1984.

subroutine *grid*(*ifreq*,*ir*)

include `'param.h'`

⟨ common statements 18 ⟩

// Constants.

$pi = 3.1415927$
$eps = 1.0 \cdot 10^{-05}$

// Perform y-interpolation on reference grid.

// Interpolate first row.

do 10 $j = 1, n, nd$
$d(1,j) = dr(ir,((j-1)/nd+1))$
$u(1,j) = ur(ir,((j-1)/nd+1))$
$v(1,j) = vr(ir,((j-1)/nd+1))$
10: **continue**

do 12 $jj = 2, nr$
do 11 $j = 1,(nd-1)$
$jjj = nd*(jj-2)+(j+1)$
$d(1,jjj) = (dr(ir,jj) - dr(ir,jj-1))*y(jjj)/dyr + (yr(jj)*dr(ir, jj-1) - yr(jj-1)*dr(ir,jj))/dyr$
$u(1,jjj) = (ur(ir,jj) - ur(ir,jj-1))*y(jjj)/dyr + (yr(jj)*ur(ir, jj-1) - yr(jj-1)*ur(ir,jj))/dyr$
$v(1,jjj) = (vr(ir,jj) - vr(ir,jj-1))*y(jjj)/dyr + (yr(jj)*vr(ir, jj-1) - yr(jj-1)*vr(ir,jj))/dyr$
11: **continue**
12: **continue**

```
    // Set number of x points and define x values.
  if (ispace ≡ 0) then
      // ispace = 0, program sets subdivisions.
    do 13 j = 1, n
      dref = d(1, j) + tide(ifreq)
      if (dref < 0.001)  dref = 0.001
      call wvnum(dref, u(1, j), freqs(ifreq), k(1, j), eps, icdw, 1, j)
13: continue
    npts = 0
    sumk = 0.
    do 14 j = 1, n
      if (d(1, j) > 0.05) then
        sumk = sumk + k(1, j)
        npts = npts + 1
      endif
14: continue
    kb(1) = sumk/float(npts)
    alw = 2.*pi/kb(1)
    anw = dxr/alw
    np = ifix(5.*anw)
    if (np < 1)  np = 1
    md(ir) = min((ix − 1), np)
    if (np > (ix − 1))  write (10, 100) ir
  endif
    // ispace = 1, user specified subdivision.
  m = md(ir) + 1
  dx = dxr/float(md(ir))
  do 15 i = 1, m
    x(i) = xr(ir) + float(i − 1)*dx
15: continue
    // interpolate values on m row.
  do 16 j = 1, n, nd
    d(m, j) = dr(ir + 1, ((j − 1)/nd + 1))
```

$u(m,j) = ur(ir+1,((j-1)/nd+1))$
$v(m,j) = vr(ir+1,((j-1)/nd+1))$
16: **continue**

do 18 $jj = 2, nr$
do 17 $j = 1,(nd-1)$
$jjj = nd*(jj-2)+(j+1)$
$d(m,jjj) = (dr(ir+1,jj) - dr(ir+1,jj-1))*y(jjj)/dyr + (yr(jj)*dr(ir+1, jj-1) - yr(jj-1)*dr(ir+1,jj))/dyr$
$u(m,jjj) = (ur(ir+1,jj) - ur(ir+1,jj-1))*y(jjj)/dyr + (yr(jj)*ur(ir+1, jj-1) - yr(jj-1)*ur(ir+1,jj))/dyr$
$v(m,jjj) = (vr(ir+1,jj) - vr(ir+1,jj-1))*y(jjj)/dyr + (yr(jj)*vr(ir+1, jj-1) - yr(jj-1)*vr(ir+1,jj))/dyr$
17: **continue**
18: **continue**

// interpolate values in x-direction

do 19 $i = 2, m-1$
do 19 $j = 1, n$
$d(i,j) = (d(m,j) - d(1,j))*x(i)/dxr + (x(m)*d(1,j) - x(1)*d(m,j))/dxr$
$u(i,j) = (u(m,j) - u(1,j))*x(i)/dxr + (x(m)*u(1,j) - x(1)*u(m,j))/dxr$
$v(i,j) = (v(m,j) - v(1,j))*x(i)/dxr + (x(m)*v(1,j) - x(1)*v(m,j))/dxr$
19: **continue**

// Add in user specified grid subdivisions (read from unit $iun(2)$).

do 30 $jr = 1, nr-1$
if $(isd(ir,jr) \equiv 1)$ **then**
$js = nd*jr + (1-nd)$
$jf = js + nd$
read $(iun(2),101)\ ((d(i,j), j = js, jf), i = 1, m)$

if $(icur \equiv 1)$ **then**
read $(iun(2),101)\ ((u(i,j), j = js, jf), i = 1, m)$
read $(iun(2),101)\ ((v(i,j), j = js, jf), i = 1, m)$
endif

do 31 $i = 1, m$
do 31 $j = js, jf$
$d(i,j) = d(i,j)*dconv(iu)$

$u(i,j) = u(i,j)*dconv(iu)$

$v(i,j) = v(i,j)*dconv(iu)$

31: **continue**

end if

30: **continue**

// Add tidal offset to all rows and establish thin film.

do 20 $i = 1, m$

do 20 $j = 1, n$

$d(i,j) = d(i,j) + tide(ifreq)$

if $(d(i,j) < 0.001)$ $d(i,j) = 0.001$

20: **continue**

// Interpolation complete, return to *model*.

return

100: **format** (`'␣model␣tried␣to␣put␣more␣spaces␣than␣allowed␣in'`, `'␣grid␣block␣'`, *i3*) ;

101: **format** (20*f10.4*) ;

end

8. CON.

Subroutine calculates constants for reference grid block *ir*.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by James T. Kirby, October 1984.

subroutine *con*(*ifreq*,*ir*)

include `'param.h'`

⟨ common statements 18 ⟩

// Constants.

$eps = 1.0 \cdot 10^{-05}$
$g = 9.80621$

// Calculate constants.

do 1 $i = 1, m$
  do 1 $j = 1, n$
    call *wvnum*(*d*(*i*,*j*),*u*(*i*,*j*),*freqs*(*ifreq*),*k*(*i*,*j*),*eps*,*icdw*,*i*,*j*)
    $sig(i,j) = freqs(ifreq) - k(i,j) * u(i,j)$
    $akd = k(i,j) * d(i,j)$
    $q(i,j) = (1. + akd/(\mathsf{sinh}(akd) * \mathsf{cosh}(akd)))/2.$
    $p(i,j) = q(i,j) * g * \mathsf{tanh}(akd)/k(i,j)$
    $dd(i,j) = (\mathsf{cosh}(4. * akd) + 8. - 2. * (\mathsf{tanh}(akd)^2))/(8. * (\mathsf{sinh}(akd)^{4.}))$
    $bottomu(i,j) = g * k(i,j)/(2 * freqs(ifreq) * \mathsf{cosh}(akd))$
1: **continue**

// Calculate the dissipation term w.

call *diss*

// Calculate the mean kb on each row.

do 11 $i = 1, m$
  $npts = 0$
  $sumk = 0.$
  do 10 $j = 1, n$
    if $(d(i,j) > 0.05)$ **then**

```
            sumk = sumk + k(i,j)
            npts = npts + 1
          endif
  10: continue
      if (npts ≡ 0) then
        kb(i) = k(i,1)
      else
        kb(i) = sumk/float(npts)
      endif
11: continue

    return

  end
```

9. FDCALC.

Perform the Crank-Nicolson finite-difference calculations on grid block ir. Method is the implicit-implicit iteration used by Kirby and Dalrymple(1983).

Parameters for use in determining the minimax approximation are defined here.

60 degree minimax coefficients.

$$a0 = 0.998214$$
$$a1 = -0.854229$$
$$b1 = -0.383283$$

70 degree minimax coefficients.

$$a0 = 0.994733$$
$$a1 = -0.890065$$
$$b1 = -0.451641$$

80 degree minimax coefficients.

$$a0 = 0.985273$$
$$a1 = -0.925464$$
$$b1 = -0.550974$$

Padé coefficients (refdif1-v2.3).

$$a0 = 1$$
$$a1 = -0.75$$
$$b1 = -0.25$$

Small angle coefficients (Radder's approximation).

$$a0 = 1.$$
$$a1 = -.5$$
$$b1 = 0.0$$

Coded by James T. Kirby, October 1984, January 1992, July 1992

Note to first users of Version 2.4: there is an unexplained and odd behavior in the minimax model when waves around islands are computed. For this reason, the program distributed here has the coefficients for the Padé model.

subroutine *fdcalc*(*ifreq*,*ir*)

include `'param.h'`

⟨ common statements 18 ⟩

real *kap*, *ksth1*, *ksth2*
complex *c1*, *c2*, *c3*, *cp1*, *cp2*, *cp3*, *ci*, *damp*
complex *ac*(*iy*), *bc*(*iy*), *cc*(*iy*), *rhs*(*iy*), *sol*(*iy*)
dimension *thet*(*iy*), *urs*(*iy*)

⟨ fdcalc statement functions 10 ⟩

// Constants.

// 70 degree minimax coefficients.

// a0=0.994733 a1=-0.890065 b1=-0.451641

// Padé coefficients.

$a0 = 1.0$
$a1 = -0.75$
$b1 = -0.25$

// Additional constants.

$u2 = 1.0$
$kap = 0.78$
$gam = 0.4$
$omeg = freqs(ifreq)$
$pi = 3.1415927$
$ci = \mathsf{cmplx}(0., 1.)$

$cdamp = 0.00025$

$alphn = 0.$
$delta1 = a1 - b1$
$delta2 = 1 + 2.*a1 - 2.*b1$

// Initialize breaking index if $ir = 1$.

if ($ir \equiv 1$) **then**
  $ifilt = 0$
  do 100 $j = 1, n$
    $ibr(j) = 0$
    $wb(1, j) = \mathsf{cmplx}(0., 0.)$

100: **continue**

endif

// Solution for m grid blocks in reference block ir.

do 200 $i = 1, (m-1)$

$ip1 = i + 1$

$it = 1$

$ih = 1$

// r.h.s. of matrix equation.

$rhs(1) = \mathsf{cmplx}(0., 0.)$

do 1 $j = 2, (n-1)$

$rhs(j) = c1(i,j)*a(i,j) + c2(i,j)*a(i,j+1) + c3(i,j)*a(i,j-1) - dx*(w(i, j) + wb(1,j))*a(i,j)/2. - dx*\mathsf{cmplx}(0., 1.)*an*anl*sig(i,j)*k(i,j)*k(i, j)*dd(i,j)*(1. - \mathsf{float}(ibr(j)))*(\mathsf{cabs}(a(i,j))^{2.})*a(i,j)/2. - dx*\mathsf{cmplx}(0., 1.)*(1. - \mathsf{float}(ibr(j)))*an*(1. - anl)*sig(i,j)*((1. + f1(i,j)*k(i,j)*k(i, j)*(\mathsf{cabs}(a(i,j))^{2.})*dd(i,j))*\mathsf{tanh}(k(i,j)*d(i,j) + f2(i,j)*k(i,j)*\mathsf{cabs}(a(i, j)))/\mathsf{tanh}(k(i,j)*d(i,j)) - 1.)*a(i,j)/2.$

1: **continue**

$rhs(n) = \mathsf{cmplx}(0., 0.)$

// Return here for iterations.

2: **if** $(it \equiv 1)\ \ ii = i$

if $(it \equiv 2)\ \ ii = ip1$

// Establish boundary conditions.

if $(ibc \equiv 1)$ **then** $ksth1 = \mathbf{real}((2.*(a(i, 2) - a(i,1))/((a(i,2) + a(i,1))*dy))*\mathsf{cmplx}(0., -1.))$ $ksth2 = \mathbf{real}((2.*(a(i,n) - a(i,n-1))/((a(i,n) + a(i,n-1))*dy))*\mathsf{cmplx}(0., -1.))$

$bc(1) = \mathsf{cmplx}(1., ksth1*dy/2.)$

$cc(1) = -\mathsf{cmplx}(1., -ksth1*dy/2.)$

$bc(n) = -\mathsf{cmplx}(1., -ksth2*dy/2.)$

$ac(n) = \mathsf{cmplx}(1., ksth2*dy/2.)$

else

$bc(1) = \mathsf{cmplx}(1., 0.)$

$cc(1) = -bc(1)$

$bc(n) = \mathsf{cmplx}(1., 0.)$

$ac(n) = -bc(n)$

endif

// Calculate dissipation in rows where breaking occurs.

do 3 $j = 1, n$

if $(ibr(j) \equiv 1)$ $wb(2,j) = \mathsf{cmplx}(1., 0.)*0.15*cg(ip1, j)*(1. - (gam*d(ip1, j)/(2.*\mathsf{cabs}(a(ii,j))))^{2.})/d(ip1, j)$

if $(ibr(j) \equiv 0)$ $wb(2,j) = \mathsf{cmplx}(0., 0.)$

3: **continue**

// Coefficients for forward row.

do 4 $j = 2, (n-1)$

$ac(j) = cp3(i,j)$

$bc(j) = cp1(i,j) + (dx/2.)*(w(i+1,j) + wb(2,j)) + \mathsf{cmplx}(0., an*anl)*sig(i+1, j)*k(i+1,j)*k(i+1,j)*dd(i+1,j)*(\mathsf{cabs}(a(ii,j))^{2.})*(dx/2.) + \mathsf{cmplx}(0., an*(1. - anl))*sig(i+1,j)*(dx/2.)*((1. + f1(i+1,j)*k(i+1,j)*k(i+1, j)*(\mathsf{cabs}(a(ii,j))^{2.})*dd(i+1,j))*\mathsf{tanh}(k(i+1,j)*d(i+1,j) + f2(i+1, j)*k(i+1,j)*\mathsf{cabs}(a(ii,j)))/\mathsf{tanh}(k(i+1,j)*d(i+1,j)) - 1.)$

$cc(j) = cp2(i,j)$

4: **continue**

// Update solution one step.

call $ctrida(1, n, ac, bc, cc, rhs, sol)$

do 5 $j = 1, n$

$a(ip1, j) = sol(j)$

$sol(j) = \mathsf{cmplx}(0., 0.)$

5: **continue**

if $((it \equiv 2) \,|\, (ih \equiv 2))$ **go to** 8

// Check for start or stop of breaking in each row.

do 6 $j = 1, n$

$urs(j) = 2.*\mathsf{cabs}(a(ip1, j))$

6: **continue**

$isave1 = 0$

$isave2 = 0$

```
do 7 j = 1, n
   iset = 0
   ireset = 0
   if (((urs(j)/d(ip1, j)) > kap) ∧ (ibr(j) ≡ 0)) iset = 1
   if (iset ≡ 1) then
      ibr(j) = 1
      isave1 = 1
   end if
   if (((urs(j)/d(ip1, j)) < gam) ∧ (ibr(j) ≡ 1)) ireset = 1
   if (ireset ≡ 1) then
      ibr(j) = 0
      isave2 = 1
   end if
7: continue
   ih = 2
      // Redo initial calculation if breaking status changes.
   if ((isave1 ≡ 1) | (isave2 ≡ 1)) go to 2
8: continue
   if (it ≡ 2) go to 9
   it = 2
   go to 2
9: continue
      // For Stokes model alone (ntype ≡ 2), test to see whether Ursell parameter is too large.
   if (ntype ≡ 2) then
      do 11 j = 1, n
         urs(j) = (cabs(a(ip1, j))/d(ip1, j))/((k(ip1, j)*d(ip1, j))^2)
         if (urs(j) > 0.5) write (10, 204) urs(j), i, j
 11: continue
   end if
      // Roll back breaking dissipation coefficient at each row.
   do 12 j = 1, n
      wb(1, j) = wb(2, j)
```

12: **continue**

// Calculate reference phase function for surface plotting.

$psibar = psibar + (kb(ip1) + kb(i))*dx/2.$

// Store plotted surface if requested.

if ($fname6 \neq$ '␣') **then**

write $(8, *)$ $x(ip1)$
write $(8, *)$ (**real** ($a(ip1, j)$*cexp(cmplx(0., $psibar$))), $j = 1, n$)

endif

// Start filter if breaking is occuring.

do 13 $j = 1, n$
if ($ibr(j) \equiv 1$) $ifilt = 1$
13: **continue**

200: **continue**

// Calculate wave angles at reference grid rows. Note: angles are not well defined in a directional, multicomponent sea, or where waves become short crested. This routine was heavily modified by Raul Medina, University of Cantabria.

do 15 $j = 1, n$
if ($a(m, j) \equiv (0., 0.)$) **then**
$akx2 = 0.$
else
$akx2$ = aimag(clog($a(m, j)$))
endif
if ($a(m - 1, j) \equiv (0., 0.)$) **then**
$akx1 = 0.$
else
$akx1$ = aimag(clog($a(m - 1, j)$))
endif
if (abs($akx2 - akx1$) $> pi$) **then**
akx = sign((2.*pi − (abs($akx1$) + abs($akx2$)))/dx, $akx1$)
else
$akx = (akx2 - akx1)/dx$
endif
if ($j \neq n$) **then**

```
      if (a(m, j + 1) ≡ (0., 0.)) then
        aky2 = 0.
      else
        aky2 = aimag(clog(a(m, j + 1)))
      endif
      if (a(m, j) ≡ (0., 0.)) then
        aky1 = 0.
      else
        aky1 = aimag(clog(a(m, j)))
      endif
    else
      if (a(m, j) ≡ (0., 0.)) then
        aky2 = 0.
      else
        aky2 = aimag(clog(a(m, j)))
      endif
      if (a(m, j − 1) ≡ (0., 0.)) then
        aky1 = 0.
      else
        aky1 = aimag(clog(a(m, j − 1)))
      endif
    endif
    if (abs(aky2 − aky1) > pi) then
      aky = sign((2.*pi − (abs(aky1) + abs(aky2)))/dy, aky1)
    else
      aky = (aky2 − aky1)/dy
    endif
    thet(j) = atan2(aky, (akx + kb(m)))
    thet(j) = 180.*thet(j)/pi
15: continue

    // Print out abs(a) at grid reference points on unit iun(4).

  mm1 = m − 1
  write (10, 205) (ir + 1), mm1
  write (10, 202) x(m)/dconv(iu), psibar

    // Wave heights on height.dat.
```

write $(12, 203)$ $(2.*\mathsf{cabs}(a(m,j))/dconv(iu), j = 1, n, nd)$

// Wave angles on *angle.dat*.

write $(9, 203)$ $(thet(j), j = 1, n, nd)$

// Water depths on *depth.dat*.

write $(16, 203)$ $(d(m,j)/dconv(iu), j = 1, n, nd)$

// Bottom velocities on *bottomu.dat*.

if ($fname7 \neq$ '␣') **then**
do 16 $j = 1, n, nd$
$bottomu(m,j) = bottomu(m,j)*\mathsf{cabs}(a(m,j))$
16: **continue**
write $(17, 203)$ $(bottomu(m,j)/dconv(iu), j = 1, n, nd)$
endif

// Write out reference grid data on disk file $iun(3)$.

write $(iun(3), *)$ $x(m)/dconv(iu)$, $psibar$
write $(iun(3), *)$ $(a(m,j)/dconv(iu), j = 1, n, nd)$

// Roll back solution to first grid level.

do 201 $j = 1, n$
$a(1,j) = a(m,j)$
201: **continue**

return

202: **format** (`'␣x='`, *f10.2*, `'␣␣␣␣reference␣phase␣psibar='`, *f20.4*) ;
203: **format** (`'␣'`, 200(*f10.4*)) ;
204: **format** (`'␣'`//`'␣warning:␣Ursell␣number␣='`, *f10.4*,
`'␣encountered␣at'`, `'grid␣location'`, *i6*, `','`, *i6*/`'␣should␣be␣using␣Stokes-H\`
`edges␣model␣(ntype=1)␣due␣to␣shallow'`, `'water'`) ;
205: **format** (`'␣grid␣row␣ir='`, *i3*, `',␣␣'`, *i3*, `'␣x-direction␣subdivisions'`, `'␣used'`) ;

end

10. FDCALC statement functions.

The following code provides the statement functions used in establishing the tridiagonal matrix structure used in *fdcalc*.

$\langle$ fdcalc statement functions 10 $\rangle \equiv$

$cg(i,j) = \mathsf{sqrt}(p(i,j)*q(i,j))$

$pv(i,j) = p(i,j) - v(i,j)*v(i,j)$

$bet(i,j) = -4.*(k(i+1,j) - k(i,j))/(dx*((k(i+1,j) + k(i,j))^2)) - 4.*(k(i+1,j)*(p(i+1,j) - u(i+1,j)^2) - k(i,j)*(p(i,j) - u(i,j)^2))/(dx*((k(i+1,j) + k(i,j))^{2.})*(p(i+1,j) + p(i,j) - (u(i+1,j)^2 + u(i,j)^2)))$

$dv(i,j) = (cg(i+1,j) + u(i+1,j))/sig(i+1,j) - (cg(i,j) + u(i,j))/sig(i,j) - delta1*dx*((v(i+1,j+1)/sig(i+1,j+1)) + (v(i,j+1)/sig(i,j+1)) - (v(i+1,j-1)/sig(i+1,j-1)) - (v(i,j-1)/sig(i,j-1)))/(2.*dy)$

$damp(i,j) = 2.*ci*cdamp*((cg(i+1,j) + u(i+1,j)) + (cg(i,j) + u(i,j)))/(dy*dy*(k(i+1,j)^2 + k(i,j)^2))$

$deltap(i,j) = a1 - b1*kb(i)/k(i,j)$

$cp1(i,j) = (cg(i+1,j) + u(i+1,j))*\mathsf{cmplx}(1., dx*(kb(i+1) - a0*k(i+1,j))) + \mathsf{cmplx}(1., 0.)*(cg(i,j) + u(i,j) + dv(i,j)*(sig(i+1,j) + sig(i,j))/4.) + 2.*omeg*\mathsf{cmplx}(0., 1.)*(-b1)*bet(i,j)*(u(i+1,j) + u(i,j))/sig(i+1,j) + 4.*omeg*(-b1)*\mathsf{cmplx}(0., 1.)*(3.*(u(i+1,j) - u(i,j))/dx + (v(i+1,j+1) + v(i,j+1) - v(i+1,j-1) - v(i,j-1))/(4.*dy))/(sig(i+1,j)*(k(i+1,j) + k(i,j))) + \mathsf{cmplx}(-2.*(-b1)/(dy*dy*(k(i+1,j) + k(i,j))) + b1*bet(i,j)*dx/(2.*dy*dy), -deltap(i,j)*dx/(2.*dy*dy))*(pv(i+1,j+1) + 2.*pv(i+1,j) + pv(i+1,j-1))/sig(i+1,j) - \mathsf{cmplx}(1., 0.)*omeg*delta2*(3.*u(i+1,j) + u(i,j))/(2.*sig(i+1,j)) + ci*omeg*(a0 - 1.)*k(i+1,j)*u(i+1,j)*dx/sig(i+1,j) + 2.*ifilt*damp(i,j) + \mathsf{cmplx}(1., 0.)*alphn*dx$

$cp2(i,j) = \mathsf{cmplx}((-delta1*dx)*(v(i+1,j) + v(i,j))/(2.*dy) + b1*u2*bet(i,j)*(u(i+1,j)*v(i+1,j) + u(i,j)*v(i,j))/(dy*sig(i+1,j+1)), (-delta1*u2)*(u(i+1,j+1)*v(i+1,j+1) + u(i,j+1)*v(i,j+1) + 2.*u(i+1,j)*v(i+1,j))/(2.*dy*sig(i+1,j+1)) + dx*(-b1)*bet(i,j)*(sig(i+1,j)*v(i+1,j) + sig(i,j)*v(i,j))/(2.*dy*sig(i+1,j+1))) + \mathsf{cmplx}(2.*(-b1)/(dy*dy*(k(i+1,j) + k(i,j))) + (-b1)*bet(i,j)*dx/(2.*dy*dy), +deltap(i,j)*dx/(2.*dy*dy))*(pv(i+1,j+1) + pv(i+1,j))/sig(i+1,j+1) + 4.*\mathsf{cmplx}(0., 1.)*(-b1)*sig(i+1,j)*v(i+1,j)/(dy*sig(i+1,j+1)*(k(i+1,j) + k(i,j))) - ifilt*damp(i,j)$

$$cp3(i,j) = \mathsf{cmplx}(-(-delta1*dx)*(v(i+1,j)+v(i,j))/(2.*dy)+(-b1)*u2*bet(i,j)*(u(i+1,j)*v(i+1,j)+u(i,j)*v(i,j))/(dy*sig(i+1,j-1)), -(-delta1*u2)*(u(i+1,j-1)*v(i+1,j-1)+u(i,j-1)*v(i,j-1)+2.*u(i+1,j)*v(i+1,j))/(2.*dy*sig(i+1,j-1))-dx*(-b1)*bet(i,j)*(sig(i+1,j)*v(i+1,j)+sig(i,j)*v(i,j))/(2.*dy*sig(i+1,j-1)))+\mathsf{cmplx}(2.*(-b1)/(dy*dy*(k(i+1,j)+k(i,j)))+(-b1)*bet(i,j)*dx/(2.*dy*dy), -(-deltap(i,j)*dx)/(2.*dy*dy))*(pv(i+1,j)+pv(i+1,j-1))/sig(i+1,j-1)-4.*\mathsf{cmplx}(0.,1.)*(-b1)*sig(i+1,j)*v(i+1,j)/(dy*sig(i+1,j-1)*(k(i+1,j)+k(i,j)))-ifilt*damp(i,j)$$

$$c1(i,j) = \mathsf{cmplx}(cg(i+1,j)+u(i+1,j)-dv(i,j)*(sig(i+1,j)+sig(i,j))/4.,0.)+\mathsf{cmplx}(1.,-dx*(kb(i)-a0*k(i,j)))*(cg(i,j)+u(i,j))+2.*\mathsf{cmplx}(0.,1.)*omeg*(-b1)*bet(i,j)*(u(i+1,j)+u(i,j))/sig(i,j)+4.*\mathsf{cmplx}(0.,1.)*omeg*(-b1)*(3.*(u(i+1,j)-u(i,j))/dx+(v(i+1,j+1)+v(i,j+1)-v(i+1,j-1)-v(i,j-1))/(4.*dy))/(sig(i,j)*(k(i+1,j)+k(i,j)))+\mathsf{cmplx}(2.*b1/(dy*dy*(k(i+1,j)+k(i,j)))-b1*bet(i,j)*dx/(2.*dy*dy), +deltap(i,j)*dx/(2.*dy*dy))*(pv(i,j+1)+2.*pv(i,j)+pv(i,j-1))/sig(i,j)-\mathsf{cmplx}(1.,0.)*omeg*delta2*(3.*u(i+1,j)+u(i,j))/(2.*sig(i,j))-ci*omeg*(a0-1.)*k(i,j)*u(i,j)*dx/sig(i,j)+2.*ifilt*damp(i,j)-\mathsf{cmplx}(1.,0.)*alphn*dx$$

$$c2(i,j) = \mathsf{cmplx}(delta1*dx*(v(i+1,j)+v(i,j))/(2.*dy)+b1*u2*bet(i,j)*(u(i+1,j)*v(i+1,j)+u(i,j)*v(i,j))/(dy*sig(i,j+1)), (-delta1*u2)*(u(i+1,j+1)*v(i+1,j+1)+u(i,j+1)*v(i,j+1)+2.*u(i,j)*v(i,j))/(2.*dy*sig(i,j+1))+4.*(-b1)*sig(i,j)*v(i,j)/(dy*(k(i+1,j)+k(i,j))*sig(i,j+1))-dx*(-b1)*bet(i,j)*(sig(i+1,j)*v(i+1,j)+sig(i,j)*v(i,j))/(2.*dy*sig(i,j+1)))+\mathsf{cmplx}(2.*(-b1)/(dy*dy*(k(i+1,j)+k(i,j)))+b1*bet(i,j)*dx/(2.*dy*dy), (-deltap(i,j)*dx)/(2.*dy*dy))*(pv(i,j+1)+pv(i,j))/sig(i,j+1)-ifilt*damp(i,j)$$

$$c3(i,j) = \mathsf{cmplx}((-delta1*dx)*(v(i+1,j)+v(i,j))/(2.*dy)-b1*u2*bet(i,j)*(u(i+1,j)*v(i+1,j)+u(i,j)*v(i,j))/(dy*sig(i,j-1)), (delta1*u2)*(u(i+1,j-1)*v(i+1,j-1)+u(i,j-1)*v(i,j-1)+2.*u(i,j)*v(i,j))/(2.*dy*sig(i,j-1))-4.*(-b1)*sig(i,j)*v(i,j)/(dy*(k(i+1,j)+k(i,j))*sig(i,j-1))+dx*(-b1)*bet(i,j)*(sig(i+1,j)*v(i+1,j)+sig(i,j)*v(i,j))/(2.*dy*sig(i,j-1)))+\mathsf{cmplx}(-2.*b1/(dy*dy*(k(i+1,j)+k(i,j)))+b1*bet(i,j)*dx/(2.*dy*dy), (-deltap(i,j)*dx)/(2.*dy*dy))*(pv(i,j)+pv(i,j-1))/sig(i,j-1)-ifilt*damp(i,j)$$

$$f1(i,j) = \tanh(k(i,j)*d(i,j))^{5.}$$

$$f2(i,j) = (k(i,j)*d(i,j)/\sinh(k(i,j)*d(i,j)))^{4.}$$

This code is used in section 9.

11. CTRIDA.

Tridiagonal matrix solution by double sweep algorithm. Present subroutine adopted from the subroutine described in:

Carnahan, Luther and Wilkes, *Applied Numerical Methods*, Wiley, 1969

The original subroutine has been modified to handle complex array coefficients and solution values. Input and output are

a, b, c	coefficients of row in tridiagonal matrix
d	right hand side vector of matrix equation
v	solution vector
ii, l	beginning and end indices of positions in the dimensioned range of the column vector.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by James T. Kirby, September 1984.

```
subroutine ctrida(ii,l,a,b,c,d,v)

  include 'param.h'

  complex a(iy), b(iy), c(iy), d(iy), v(iy), beta(iy), gamma(iy)

    // Compute intermediate vectors beta and gamma.

  beta(ii) = b(ii)
  gamma(ii) = d(ii)/beta(ii)
  iip1 = ii + 1

  do 1 i = iip1, l
    beta(i) = b(i) - a(i)*c(i - 1)/beta(i - 1)
    gamma(i) = (d(i) - a(i)*gamma(i - 1))/beta(i)
1: continue

    // Compute solution vector v.

  v(l) = gamma(l)
  last = l - ii

  do 2 k = 1, last
    i = l - k
    v(i) = gamma(i) - c(i)*v(i + 1)/beta(i)
```

```
2: continue
   return
end
```

12. DISS.

Subroutine calculates the dissipation at a single grid point based on values of the switch *iw* at that point.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by James T. Kirby, October 1984.

subroutine *diss*

include `'param.h'`

⟨ common statements 18 ⟩

real *nu*, *cp*, *kd*

// Statement function.

$sq(i,j) = \mathsf{sqrt}(nu/(2.*sig(i,j)))$

// Constants.

$nu = 1.3 \cdot 10^{-06}$
$cp = 4.5 \cdot 10^{-11}$
$g = 9.80621$
$pi = 3.1415927$

// Value of f here is value assuming $\tau = (f/8)u^2$.

// $f = 4f_w$; f_w is the wave friction factor

$f = 0.01*4.0$

do 1 $j = 1, n$
  do 1 $i = 1, m$
    $w(i,j) = \mathsf{cmplx}(0., 0.)$
    $kd = k(i,j)*d(i,j)$

// If $\mathit{iff}(1) = 1$, use turbulent boundary layer damping.

if $(\mathit{iff}(1) \equiv 1)$
$w(i,j) = 2.*f*\mathsf{cabs}(a(1,j))*sig(i,j)*k(i,j)/(\mathsf{sinh}(2.*kd)*\mathsf{sinh}(kd)*3.*pi)$

// If $\mathit{iff}(2) = 1$, add porous bottom damping.

if $(\mathit{iff}(2) \equiv 1)$ $w(i,j) = w(i,j) + (g{*}k(i,j){*}cp/(nu{*}(\mathsf{cosh}(kd)^2))){*}\mathsf{cmplx}(1.,0.)$

// If $\mathit{iff}(3) = 1$, add boundary layer damping.

if $(\mathit{iff}(3) \equiv 1)$ $w(i,j) = w(i,j) + 2.{*}k(i,j){*}sig(i,j){*}sq(i,$
$j){*}(1. + (\mathsf{cosh}(kd)^2)){*}\mathsf{cmplx}(1.,-1.)/\mathsf{sinh}(2.{*}kd)$

1: **continue**

return

end

13. RAND.

Generate a floating point pseudo random number between 0 and 1 by the multiplicative congruential method. see knuth, d.e., 1969, p. 155.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by Robert A. Dalrymple, January 1986.

```
function rand(x)
  ix = ifix(32767.*x)
  irand = mod(4573*ix + 6923, 32767)
  rand = float(irand)/32767.
  return
end
```

14. FACT.

compute the factorial of xi

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by Robert A. Dalrymple, January 1986

```
function fact(xi)
   prod = 1.
   if (xi > 1.) then
      do 17 ii = 2, int(xi)
         prod = prod*float(ii)
17:  continue
  endif
  fact = prod
  return
end
```

15. BNUM.

Compute the combination $in!/(n!(n-in)!)$.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by Robert A. Dalrymple, January 1986.

```
subroutine bnum(in,n,bn)
   xin = in
   xt = fact(xin)
   xb = fact(float(n))*fact(float(in - n))
   bn = xt/xb

   return

end
```

16. ACALC.

Calculate the normalization factor a for the directional spectrum such that $\int_{-\theta_m}^{\theta_m} \cos(\theta/2)^{2*nsp} d\theta = 1$, where, in code, $\theta_m = thmax$.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by Robert A. Dalrymple, January 1986.

```
subroutine acalc(thmax,nsp,a)
   itn = 2*nsp
   call bnum(itn,nsp,bn)
   a = thmax*bn/(2.**(itn-1))
   sum = 0.
   do 10 ik = 1, nsp
      ki = ik - 1
      call bnum(itn,ki,bn)
10: sum = sum + bn*sin(float(nsp - ki)*thmax)/float(nsp - ki)
   a = a + sum/(2.**(itn-2))
   return
end
```

17. WVNUM.

Calculate wavenumber k according to the form

$$(\sigma - ku)^2 = gk\tanh(kd)$$

where

d	local water depth
$s = \sigma$	absolute frequency
g	gravitational acceleration constant
u	x-component of ambient current
$eps = \epsilon$	tolerance for iteration convergence
i, j	indices in finite-difference grid
$icdw$	switch
	=0, no convergence failures encountered
	=1, at least one convergence failure

Solution is by Newton-Raphson iteration using Eckart's approximation as a seed value.

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware
Newark, DE 19716

Coded by James T. Kirby, September,1984

```
subroutine wvnum(d,u,s,k,eps,icdw,i,j)

  include 'param.h'

  common /ref2/ dr(ixr,iyr), ur(ixr,iyr), vr(ixr,iyr), iun(8), iinput, ioutput

  real k, kn

    // Constants.

  g = 9.806
  pi = 3.1415927
  k = s*s/(g*sqrt(tanh(s*s*d/g)))

    // Newton-Raphson iteration.

  do 1 ii = 1, 20
    f = s*s - 2.*s*k*u + k*k*u*u - g*k*tanh(k*d)
    fp = -2.*s*u + 2.*k*u*u - g*tanh(k*d) - g*k*d/(cosh(k*d)**2.)
    kn = k - f/fp
    if ((abs(kn - k)/kn) < eps) go to 2
```

```
      k = kn
1:    continue
      t = 2.*pi/(sqrt(g*k*tanh(k*d)) + k*u)
      write (10, 100) i, j, k, u, d, f, t
      icdw = 1

      return

2:    k = kn

      return

100:  format (' wavenumber iter. failed to converge on row', i10, '  column',
         i10 /'      k=', f15.8, '      u=', f15.8 /'      d=', f15.8, '      f=',
         f15.8 /'      t=', f15.8) ;

    end
```

18. COMMON STATEMENTS.

⟨ common statements 18 ⟩ ≡

common /*ref1*/ *mr*, *nr*, *ispace*, *nd*, *md*(*ixr*), *iu*, *dconv*(2), *iff*(3), *icur*, *ibc*

common /*ref2*/ *dr*(*ixr*,*iyr*), *ur*(*ixr*,*iyr*), *vr*(*ixr*,*iyr*), *iun*(8), *iinput*, *ioutput*

common /*ref3*/ *dxr*, *dyr*, *xr*(*ixr*), *yr*(*iyr*), *x*(*ix*), *y*(*iy*)

common /*ref4*/ *isd*(*ixr*,*iyr*)

common /*block1*/ *d*(*ix*,*iy*), *u*(*ix*,*iy*), *v*(*ix*,*iy*), *m*, *n*, *dx*, *dy*, *ibr*(*iy*)

common /*con1*/ *q*(*ix*,*iy*), *p*(*ix*,*iy*), *sig*(*ix*,*iy*), *bottomu*(*ix*,*iy*)

common /*con2*/ *k*(*ix*,*iy*), *kb*(*ix*), *w*(*ix*,*iy*), *dd*(*ix*,*iy*), *wb*(2, *iy*)

common /*nlin*/ *an*, *anl*, *ntype*

common /*wav1*/ *iwave*, *nfreqs*, *freqs*(*ncomp*), *edens*(*ncomp*), *nwavs*(*ncomp*)

common /*wav2*/ *amp*(*ncomp*,*ncomp*), *dir*(*ncomp*,*ncomp*), *tide*(*ncomp*), *seed*, *thet0*

common /*comp*/ *a*(*ix*,*iy*), *psibar*, *ifilt*

common /*names*/ *fname1*, *fname2*, *fname3*, *fname4*, *fname5*, *fname6*, *fname7*, *fname8*, *fname9*, *fname10*, *fname11*, *fname12*, *fname13*, *fname14*, *fname15*, *fnamein*

real *k*, *kb*

complex *w*, *a*, *wb*

character∗255 *fname1*, *fname2*, *fname3*, *fname4*, *fname5*, *fname6*, *fname7*, *fname8*, *fname9*, *fname10*, *fname11*, *fname12*, *fname13*, *fname14*, *fname15*, *fnamein*

This code is used in sections 1, 2, 5, 6, 7, 8, 9, and 12.

6 Appendix B: Fortran Codes for Generating and Post-Processing Data Files.

This appendix provides listings for the following programs:

- *infile1.f* - code fragment linked to *refdif1* during compiling; used to specify *indat.dat* file name.
- *indat-createv25.f* - used to create the *indat.dat* file based solely on input from the user.
- *indat-convertv25.f* - used to convert old (Version 2.4 or earlier *indat.dat—item* indat-convert.f - *used to convert old (Version 2.4 or earlier indat.dat* files to new (Version 2.5) *indat.dat* files.
- *datgenv25.f* - used to create Version 2.5 *indat.dat* and *refdat.dat* files for specific examples.
- *surface.f* - used to convert the data in file *surface.dat* to a regularly spaced, ascii formatted array representing an instantaneous picture of the water surface.
- *refdifplot.m* - sample Matlab program illustrating the reading and plotting of the output data.

Note that the versions of the programs supplied with the program distribution may be slightly updated relative to the codes listed here.

6.1 *infile1.f*

```
c*--------------------------------------------------------------------------
c*  infile1
c*
c*  Provide code for defining the indat.dat file name for refdif 1 and
c*  refdif s.
c*
c*  James T. Kirby, October 10, 1994.
c*
c*--------------------------------------------------------------------------

      subroutine infile(fnamein)
      character*255 fnamein
      fnamein='indat.dat'

      return
      end
```

6.2 *indat-createv25.f*

```
c*-----------------------------------------------------------------------------
c*    indat-createv25.f
c*
c*    This program generates an indat.dat file by asking the operator a
c*    series of questions.  This file is intended to make life a little
c*    easier - its function is just as easily carried out manually if you
c*    are used to the form of the indat.dat file.
c*
c*    James T. Kirby
c*    Center for Applied Coastal Research
c*    University of Delaware
c*    Newark, DE 19716
c*
c*    (302) 831-2438, FAX (302) 831-1228, kirby@coastal.udel.edu
c*
c*    Last revision 12/22/94.
c*
c*-----------------------------------------------------------------------------

      program indatcreate

      include 'param.h'

      dimension md(ixr), dconv(2), iff(3)
      dimension freqs(ncomp), edens(ncomp), nwavs(ncomp)
      dimension amp(ncomp,ncomp), dir(ncomp,ncomp), tide(ncomp)

      character*255 fname1,fname2,fname3,fname4,fname5,fname6,fname7,
     1             fname8,fname9,fname10,fname11,fname12,fname13,
     1              fname14,fname15,fnamein

      data fname1 /'refdat.dat'/, fname2 /'outdat.dat'/,
     1     fname3/'subdat.dat'/,fname4/'wave.dat'/,
     1     fname5/'owave.dat'/,fname6/'surface.dat'/,
     1     fname7/'bottomu.dat'/,fname8/'angle.dat'/,
     1     fname9/' '/,fname10/'refdif1.log'/,
     1     fname11/'height.dat'/,fname12/'sxx.dat'/,
     1     fname13/'sxy.dat'/,fname14/'syy.dat'/,
     1     fname15/'depth.dat'/


      namelist /ingrid/ mr, nr, iu, ntype, icur, ibc, dxr, dyr, dt,
     1                  ispace, nd, iff, isp, iinput, ioutput
     1         /inmd/  md
     1         /fnames/ fname1,fname2,fname3,fname4,fname5,fname6,
     1                  fname7,fname8,fname9,fname10,fname11,fname12,
     1                  fname13,fname14,fname15
     1         /waves1a/ iwave, nfreqs
     1         /waves1b/ freqs, tide, nwavs, amp, dir
     1         /waves1c/ thet0, freqs, tide, edens, nwavs, nseed
     1         /waves2/ freqin, tidein
```

```
      call infile(fnamein)

      open(unit=10,file=fnamein)

      write(*,*)' enter name for .dat file containing reference grid in',
     1' single quotes'
      read(*,*) fname1

      write(*,*)' enter name for ouput data file'
      read(*,*) fname2

      write(10,nml=fnames)

c  Enter control data.

      write(*,*)' enter grid dimensions mr, nr'
      read(*,*) mr,nr

      write(*,*)' enter grid spacings dxr, dyr and depth tolerance dt'
      read(*,*) dxr, dyr, dt

      write(*,*)' input iu: 1=mks, 2=english'
      read(*,*) iu

      write(*,*)' input dispersion relationship; ntype: 0=linear,'
      write(*,*)'                                   1=composite, 2=stokes'
      read(*,*)ntype

      write(*,*)' input lateral boundary condition; ibc: 0=closed'
      write(*,*)'                                                1=open'
      read(*,*) ibc

      write(*,*)' input ispace (0=program picks x spacing,',
     1'  1=user chooses)'
      read(*,*) ispace

      write(*,*)' input nd (# y divisions, 1 is minimum)'
      read (*,*) nd

      if(ispace.eq.0) go to 105

      write(*,*)' constant or variable x spacing?(0 for constant)'
      read(*,*) ians1

      if(ians1.eq.0) then
      write(*,*) ' input constant md'
      read(*,*) mdc
      do 103 iko=1,mr-1
      md(iko)=mdc
 103  continue
      else
      write(*,104)
```

```
 104  format(' input md(i) for i=1 to mr-1')
      read(*,*) (md(i),i=1,mr-1)
      endif

 105  write(*,106)
 106  format(' input if(1) turbulent, if(2) porous, if(3) laminar')
      write(*,*) '  standard  choice:  1, 0, 0'
      read(*,*) iff(1), iff(2), iff(3)

      write(*,*)' input isp (subgrid features) :standard  0'
      read(*,*) isp

      write(*,108)
 108  format(' input values of iinput, ioutput:'/
     1' iinput: 1 standard, i.e., not starting from previous run'/
     1'         2 if starting from previous run'/
     1' ioutput: 1 standard, not saving restart data'/
     1'          2 if saving restart data')
      read(*,*)iinput,ioutput

      write(*,115)
 115  format(' input value of isurface:'/
     1'  isurface = 0:   no surface picture generated'/
     1'  isurface = 1:   surface picture generated')
      read(*,*) isurface

      if(isurface.eq.0) fname6 = ' '

      write(10,nml=ingrid)

      if(ispace.eq.1) write(10,nml=inmd)

      if(iinput.eq.1) then
c*-----------------------------------------------------------------------
c*    write waves1 portion of indat.dat
c*-----------------------------------------------------------------------
      write(*,*)' input iwave (1 discrete, 2 directional spread)'
      read(*,*) iwave

      write(*,*) ' input nfreq (# of frequencies)'
      read(*,*) nfreqs

      write(10, nml=waves1a)

      if (iwave.eq.2) then
      write(*,*)' enter central direction thet0'
      read(*,*) thet0
      endif

      do 113 ifreq=1,nfreqs
c*-----------------------------------------------------------------------
c*    line 10, iinput=1
c*-----------------------------------------------------------------------
      write(*,109)
```

```
 109  format(' input wave period and tide stage')
      read(*,*) freqs(ifreq), tide(ifreq)

c*-----------------------------------------------------------------------
c*    line 11, iwave=1, iinput=1
c*-----------------------------------------------------------------------
      if(iwave.eq.1) then

      write(*,110)
 110  format(' input # of waves per frequency, nwavs')
      read(*,*) nwavs(ifreq)

c*-----------------------------------------------------------------------
c*    line 12, iwave=1, iinput=1
c*-----------------------------------------------------------------------
      do 111 iwavs=1,nwavs(ifreq)
      write(*,*)' input amplitude and direction'
      read(*,*) amp(ifreq,iwavs), dir(ifreq,iwavs)

 111  continue

      else

c*----------------------------------------------------------------------
c*    iwave=2, iinput=1
c*----------------------------------------------------------------------
      write(*,112)
 112  format('input en. density and on next line, directional',
     1'  spreading factor')
      read(*,*) edens(ifreq)
      read(*,*) nwavs(ifreq)

      nseed=500

      endif
 113  continue

      if (iwave .eq. 1) write(10, nml=waves1b)
      if (iwave .eq. 2) write(10, nml=waves1c)

      endif

      if ( iinput .eq. 2) then
c*----------------------------------------------------------------------
c*    line 9, iinput=2
c*----------------------------------------------------------------------
      write(*,*)' input wave period and tide stage'
      read(*,*) freqin ,tidein

      write(10, nml=waves2)

      endif

      close(10)
```

```
stop
end
```

6.3 *indat-convertv25.f*

```
c*-----------------------------------------------------------------------
c*    indat-convertv25.f
c*
c*    This program reads an old (version 2.4 or earlier) indat.dat file
c*    for ref/dif 1 and converts it to a new (version 2.5 or later) file
c*    with the temporary file name indat.new.  The new file should be
c*    renamed to indat.dat.
c*
c*    James T. Kirby
c*    Center for Applied Coastal Research
c*    University of Delaware
c*    Newark, DE 19716
c*
c*    (302) 831-2438, FAX (302) 831-1228, kirby@coastal.udel.edu
c*
c*    Last revision 12/22/94.
c*
c*-----------------------------------------------------------------------

      include 'param.h'

      dimension md(ixr), dconv(2), iff(3), iun(8)
      dimension freqs(ncomp), edens(ncomp), nwavs(ncomp)
      dimension amp(ncomp,ncomp), dir(ncomp,ncomp), tide(ncomp)

      character*255 fname1,fname2,fname3,fname4,fname5,fname6,fname7,
     1              fname8,fname9,fname10,fname11,fname12,fname13,
     1              fname14,fname15,fnamein


      data fname1 /'refdat.dat'/, fname2 /'outdat.dat'/,
     1     fname3/'subdat.dat'/,fname4/'wave.dat'/,
     1     fname5/'owave.dat'/,fname6/'surface.dat'/,
     1     fname7/'bottomu.dat'/,fname8/'angle.dat'/,
     1     fname9/' '/,fname10/'refdif1.log'/,
     1     fname11/'height.dat'/,fname12/'sxx.dat'/,
     1     fname13/'sxy.dat'/,fname14/'syy.dat'/,
     1     fname15/'depth.dat'/


      namelist /ingrid/ mr, nr, iu, ntype, icur, ibc, dxr, dyr, dt,
     1                  ispace, nd, iff, isp, iinput, ioutput
     1         /inmd/  md
     1         /fnames/ fname1,fname2,fname3,fname4,fname5,fname6,
     1                  fname7,fname8,fname9,fname10,fname11,fname12,
     1                  fname13,fname14,fname15
     1         /waves1a/ iwave, nfreqs
     1         /waves1b/ freqs, tide, nwavs, amp, dir
     1         /waves1c/ thet0, freqs, tide, edens, nwavs, nseed
     1         /waves2/ freqin, tidein
```

```
c  Define device number |iun(5)| for reference grid input.
c  This line must be set during program installation, prior to compilation.
c  Also construct an |open| statement if needed.

      call infile(fnamein)

      iun(5)=5
      iun(6)=6
      open(unit=iun(5),file=fnamein,status='old')
      open(unit=iun(6),file='indat.new')

      write(iun(6),nml=fnames)

c  Read unit numbers from indat.dat.  These are not used in indat.new.

      read(iun(5),*)(iun(i),i=1,3)

c  Read control data from unit |iun(5)|.

      read(iun(5),*) mr,nr
      read(iun(5),*) iu, ntype, icur, ibc
      read(iun(5),*) dxr, dyr, dt
      read(iun(5),*) ispace, nd

      if(ispace.eq.1) then
      read(iun(5),*) (md(i),i=1,mr-1)
      endif

      read(iun(5),*) (iff(i),i=1,3)

      read(iun(5),*) isp

      read(iun(5),*) iinput,ioutput

      if (iinput .eq. 1) then

c  Read |iwave|, |nfreqs|.

      read(iun(5),*) iwave, nfreqs

      if(iwave.eq.2) then
      read(iun(5),*)thet0
      endif

c  For each frequency, enter the wave period and tidal offset.

      do 3 ifreq=1,nfreqs
      read(iun(5),*) freqs(ifreq), tide(ifreq)


c  If |iwave = 1|, read the number of discrete components.

      if(iwave.eq.1) then
      read(iun(5),*) nwavs(ifreq)
```

```
      do 1 iwavs=1,nwavs(ifreq)
      read(iun(5),*) amp(ifreq,iwavs), dir(ifreq,iwavs)
 1    continue
      endif

c  If |iwave = 2|, read the parameters for each frequency.

      if(iwave.eq.2)then
      read(iun(5),*) edens(ifreq)
      read(iun(5),*)  nwavs(ifreq),nseed
      endif

 3    continue

      endif

c  If |iinput = 2|, read in wave period and tidal offset.

      if(iinput.eq.2)then
      nfreqs=1
      read(iun(5),*) freqin, tidein
      endif

      write(iun(6),nml=ingrid)

      if (ispace.eq.1) write(iun(6),nml=inmd)

      if (iinput .eq. 1) then

        write(iun(6),nml=waves1a)
        if (iwave .eq. 1) write (iun(6), nml = waves1b)
        if (iwave .eq. 2) write (iun(6), nml = waves1c)

      endif

      if (iinput .eq. 2) write (iun(6), nml = waves2)

      close(iun(6))



      stop


      end
```

6.4 *datgenv25.f*

```
C*-------------------------------------------------------------------
C*
C*    datgenv25.f
C*
C*    This program generates input data files for several example
c*    applications of REF/DIF 1.  In particular, the first three cases
c*    listed here correspond to the three test cases shown in the
c*    User's Manual.
c*
c*    James T. Kirby
c*
c*    kirby@coastal.udel.edu, (302) 831-2438, FAX (302) 831-1228
c*
c*    Center for Applied Coastal Research
c*    Department of Civil Engineering
c*    University of Delaware
c*    Newark, DE 19716
C*
C*    January 1991, revised July 1994 for REF/DIF 1 version 2.5.
c*
c*    Last revision 12/22/94.
C*
C*-------------------------------------------------------------------

      include 'param.h'

      dimension iun(4), md(ixr)

      common/ref/ dr(ixr,iyr),ur(ixr,iyr),vr(ixr,iyr),mr,nr,dxr,dyr,
     1itype
      common/ind/ iu,ntype,icur,ibc,ispace,nd,iff,isp,iinput,
     1iwave,nfreqs,freqs,tide,nwavers,amp,dir,edens
      common/dims/ x(ixr),y(iyr)
      dimension iff(3)
      dimension amp(ncomp,ncomp),dir(ncomp,ncomp),tide(ncomp),
     1          freqs(ncomp),edens(ncomp),nwavs(ncomp)
      character*255 fname1,fname2,fname3,fname4,fname5,fname6,fname7,
     1              fname8,fname9,fname10,fname11,fname12,fname13,
     1              fname14,fname15,fnamein

      data fname1/'refdat.dat'/, fname2/'outdat.dat'/,
     1     fname3/'subdat.dat'/, fname4/'wave.dat'/,
     1     fname5/'owave.dat'/, fname6/'surface.dat'/,
     1     fname7/'bottomu.dat'/,fname8/'angle.dat'/,
     1     fname9/' '/,fname10/'refdif1.log'/,
     1     fname11/'height.dat'/,fname12/'sxx.dat'/,
     1     fname13/'sxy.dat'/,fname14/'syy.dat'/,
     1     fname15/'depth.dat'/


      namelist /ingrid/ mr, nr, iu, ntype, icur, ibc, dxr, dyr, dt,
```

```
     1                  ispace,nd,iff,isp,iinput,ioutput
     1         /inmd/ md
     1         /fnames/ fname1,fname2,fname3,fname4,fname5,fname6,
     1                  fname7,fname8,fname9,fname10,fname11,fname12,
     1                  fname13,fname14,fname15
     1         /waves1a/ iwave, nfreqs
     1         /waves1b/ freqs, tide, nwavs, amp, dir
     1         /waves1c/ thet0, freqs, tide, edens, nwavs, nseed
     1         /waves2/ freqin, tidein
c*--------------------------------------------------------------------
c*    setup the logical devices for input:
c*          *=keyboard input
c*          iun(2)=output file "refdat.dat"
c*          iun(3)=screen output (use 0 for sun, 3 for pc)
c*          iun(4)=output file "indat.dat"
c*---------------------------------------------------------------------
      iun(2)=20
      iun(3)=0

      call infile(fnamein)

      iun(4)=24
      open(iun(4),file=fnamein)

c*---------------------------------------------------------------------
c*    initialize all entries for indat.dat prior to generating the
c*    depth grid.
c*---------------------------------------------------------------------
      iu=0
      ntype=0
      icur=0
      ibc=0
      ispace=0
      nd=1
      if1=0
      if2=0
      if3=0
      isp=0
      iinput=0
      iwave=0
      nfreqs=0

      nwaves=0

c*---------------------------------------------------------------------
c*    open file for reference grid data
c*---------------------------------------------------------------------

      open(iun(2),file=fname1)

c*---------------------------------------------------------------------
c*    establish depth grid
c*---------------------------------------------------------------------
```

```
      call depth(iun)

c*-------------------------------------------------------------------
c*    calculate constants
c*-------------------------------------------------------------------
      dt=10.
      call con
c*-------------------------------------------------------------------
c*    write reference grid data
c*-------------------------------------------------------------------
      do 1 i=1,mr
      write(iun(2),100)(dr(i,j),j=1,nr)
 1    continue

      if(icur.eq.1)then

      do 2 i=1,mr
      write(iun(2),100)(ur(i,j),j=1,nr)
 2    continue

      do 3 i=1,mr
      write(iun(2),100)(vr(i,j),j=1,nr)
 3    continue

      endif

      close(iun(2))

 100  format(20f10.4)

c*---------------------------------------------------------------------
c*    generation of file indat.dat
c*---------------------------------------------------------------------

      write(*,*) ' do you want to create indat.dat? yes=1'
      read(*,*) ians

      if(ians.eq.1) then
      open(iun(4),file='indat.dat')

c*---------------------------------------------------------------------
c*    write fnames portion of namelist file
c*---------------------------------------------------------------------

      write(iun(4),nml=fnames)

c*---------------------------------------------------------------------
c*    write ingrid portion of namelist file
c*---------------------------------------------------------------------

      if(iu.eq.0)then
      write(iun(3),*)' input iu: 1=mks, 2=english'
      read(*,*) iu
      endif
```

```
      write(iun(3),*)' input dispersion relationship; ntype: 0=linear,'
      write(iun(3),*)'                                 1=composite, 2=stokes'
      read(*,*)ntype
      write(iun(3),*)' input lateral boundary condition; ibc: 0=closed'
      write(iun(3),*)'                                                1=open'
      read(*,*) ibc

      write(iun(3),102)
102   format(' input ispace (0=program picks x spacing, 1=user choses)')
      read(*,*) ispace
      write(*,*)' input nd (# y divisions, 1 is minimum)'
      read (*,*) nd

      if(ispace.eq.0) go to 105
      write(iun(3),*)' constant or variable x spacing?(0 for constant)'
      read(*,*) ians1
      if(ians1.eq.0) then
      write(*,*) ' input constant md'
      read(*,*) mdc
      do 103 iko=1,mr-1
      md(iko)=mdc
103   continue
      else
      write(iun(3),104)
104   format(' input md(i) for i=1 to mr-1')
      read(*,*) (md(i),i=1,mr-1)
      endif

105   write(iun(3),106)
106   format(' input if(1) turbulent, if(2) porous, if(3) laminar')
      write(iun(3),*) '  standard  choice:  1, 0, 0'
      read(*,*) iff(1), iff(2), iff(3)

      write(iun(3),107)
107   format(' input isp (subgrid features) :standard  0')
      read(*,*) isp

      write(iun(3),108)
108   format(' input values of iinput, ioutput:'/
     1' iinput: 1 standard, i.e., not starting from previous run'/
     1'         2 if starting from previous run'/
     1' ioutput: 1 standard, not saving restart data'/
     1'          2 if saving restart data')
      read(*,*)iinput,ioutput

      write(iun(3),115)
115   format(' input value of isurface:'/
     1'  isurface = 0:   no surface picture generated'/
     1'  isurface = 1:   surface picture generated')
      read(*,*) isurface
      if(isurface.eq.0) fname6 = ' '

      write(iun(4),nml=ingrid)
```

```
      if(ispace.eq.1) write(iun(4),nml=inmd)

      if(iinput.eq.1) then
c*-----------------------------------------------------------------------
c*    write waves1 portion of indat.dat
c*-----------------------------------------------------------------------
      write(iun(3),*)' input iwave (1 discrete, 2 directional spread)'
      read(*,*) iwave
      write(iun(3),*) ' input nfreq (# of frequencies)'
      read(*,*) nfreqs

      write(iun(4), nml=waves1a)

      if (iwave.eq.2) then
      write(*,*)' enter central direction thet0'
      read(*,*) thet0
      endif

      do 113 ifreq=1,nfreqs
c*-----------------------------------------------------------------------
c*    line 10, iinput=1
c*-----------------------------------------------------------------------
      write(iun(3),109)
 109  format(' input wave period and tide stage')
      read(*,*) freqs(ifreq), tide(ifreq)

c*-----------------------------------------------------------------------
c*    line 11, iwave=1, iinput=1
c*-----------------------------------------------------------------------
      if(iwave.eq.1) then

      write(iun(3),110)
 110  format(' input # of waves per frequency, nwavs')
      read(*,*) nwavs(ifreq)

c*-----------------------------------------------------------------------
c*    line 12, iwave=1, iinput=1
c*-----------------------------------------------------------------------
      do 111 iwavs=1,nwavs(ifreq)
      write(iun(3),*)' input amplitude and direction'
      read(*,*) amp(ifreq,iwavs), dir(ifreq,iwavs)

 111  continue

      else

c*----------------------------------------------------------------------
c*    iwave=2, iinput=1
c*----------------------------------------------------------------------
      write(iun(3),112)
 112  format('input en. density and on next line, directional',
     1'  spreading factor')
      read(*,*) edens(ifreq)
      read(*,*) nwavs(ifreq)
```

```
      nseed=500

      endif
 113  continue

      if (iwave .eq. 1) write(iun(4), nml=waves1b)
      if (iwave .eq. 2) write(iun(4), nml=waves1c)

      endif

      if ( iinput .eq. 2) then
c*----------------------------------------------------------------------
c*    line 9, iinput=2
c*----------------------------------------------------------------------
      write(iun(3),*)' input wave period and tide stage'
      read(*,*) freqin ,tidein

      write(iun(4), nml=waves2)

      endif

      close(iun(4))

      endif

      stop
      end

c*----------------------------------------------------------------------
      subroutine con
c*----------------------------------------------------------------------

      include 'param.h'

      common/ref/ d(ixr,iyr),u(ixr,iyr),v(ixr,iyr),m,n,dx,dy,itype
      common/ind/ iu,ntype,icur,ibc,ispace,nd,iff,isp,iinput,
     1iwave,nfreqs,freqs,tide,nwavers,amp,dir,edens
      common/dims/ x(ixr),y(iyr)
      dimension iff(3)
      dimension amp(ncomp,ncomp),dir(ncomp,ncomp),tide(ncomp),
     1          freqs(ncomp),edens(ncomp),nwavs(ncomp)

      do 1 i=1,m
      do 1 j=1,n
      if((icur.eq.1).and.(itype.eq.3))then
      xp=x(m)-x(i)
      u(i,j)=-0.02295*exp(-((xp/76.2)**2)/2.)*exp(-((y(j)/7.62)**2)
     1/2.)*xp
      v(i,j)=-0.2188*(2.-(xp/76.2)**2)*exp(-((xp/76.2)**2)/2.)*
     1erfjk(abs(y(j))/107.76)*y(j)/abs(y(j))
      else
      u(i,j)=0.
      v(i,j)=0.
```

```
      endif
 1    continue

      return
      end

c*-----------------------------------------------------------------------
      subroutine depth(iun)
c*-----------------------------------------------------------------------

      include 'param.h'

      common/ref/ d(ixr,iyr),u(ixr,iyr),v(ixr,iyr),m,n,dx,dy,itype
      common/ind/ iu,ntype,icur,ibc,ispace,nd,iff(3),isp,iinput,
     1iwave,nfreqs,freqs,tide,nwavers,amp,dir,edens
      common/dims/ x(ixr),y(iyr)
      dimension iun(3)
      dimension amp(ncomp,ncomp),dir(ncomp,ncomp),tide(ncomp),
     1          freqs(ncomp),edens(ncomp),nwavs(ncomp)


      write(iun(3),1)
  1   format(' ***********  parabolic model in rectangular',
     1 ' grid  **********'//
     2 ' input type of bottom desired'//
     3 ' 1=surface piercing island'/
     4 ' 2=bbr, submerged shoal'/
     5 ' 3=arthur rip current'/
     6 ' 4=test case, planar bottom'/
     7 ' 5=radder(1979), configuration 2'/
     8 ' 6=grazing incidence on linear caustic'/
     9 ' 7=whalin''s channel'/
     1 ' 8=surface piercing breakwater'/
     2 ' 9=channel'/
     3 '10=breakwater')
      read (*,*) itype

      if ( itype .eq. 1) then
c*-------------------------------------------------------------------
c*    surface piercing island
c*-------------------------------------------------------------------
      write(iun(3),101)
 101  format(' surface piercing island')
      write(iun(3),102)
 102  format(' input m,n,dx,dy,depth,period')
      read (*,*) m,n,dx,dy,dep,t
      write(iun(3),*)m,n,dx,dy,dep,t
      write(iun(3),103)
 103  format(' input crest height, x semiaxis, y semiaxis')
      read (*,*) hb, xa, ya
      write (iun(3),*)hb,xa,ya
      sig=2.*3.1415927/t
      do 104 i=1,m
      x(i)=float(i-1)*dx
```

```
 104  continue
      do 105 j=1,n
      y(j)=(float(j-1)-0.5)*dy
 105  continue
      xc=xa+3.*dx
      do 106 i=1,m
      do 106 j=1,n
      d(i,j)=dep-(1.-sqrt((((x(i)-xc)/xa)**2)+((y(j)/ya)**2)
     1))*hb
      if(d(i,j).gt.dep)d(i,j)=dep
 106  continue
      end if

      if ( itype .eq. 2) then
c*----------------------------------------------------------------------
c*    bbr, submerged shoal
c*----------------------------------------------------------------------
      iu=1
      m=100
      n=100
      c20=cos(20.*3.1415927/180.)
      s20=sin(20.*3.1415927/180.)
      dx=0.25
      dy=0.25
      do 151 i=1,m
 151  x(i)=float(i-1)*dx
      do 152 j=1,n
 152  y(j)=float(j-1)*dy
      do 154 i=1,m
      do 154 j=1,n
      xp=(x(i)-10.5)*c20-(y(j)-10.)*s20
      yp=(x(i)-10.5)*s20+(y(j)-10.)*c20
      test=((yp/4.)**2)+((xp/3.)**2)
      if(xp.lt.(-5.84))d(i,j)=0.45
      if(xp.ge.(-5.84))d(i,j)=0.45-0.02*(xp+5.84)
      if(test.gt.1) go to 153
      d(i,j)=d(i,j)-(0.5*sqrt(1.-((yp/5.)**2)-((xp/3.75)**2))
     1-0.3)
 153  continue
 154  continue
      end if

      if ( itype .eq. 3) then
c*---------------------------------------------------------------------
c*    arthur (1952) rip current
c*---------------------------------------------------------------------
      iu=1
      icur=1
      dx=5.0
      dy=5.0
      slope=0.02
      m=100
      n=100
      do 201 i=1,m
```

```
      x(i)=float(i-1)*dx
 201  continue
      do 202 j=1,n
      y(j)=float(2*j-n-1)*dy/2.
 202  continue
      xm=x(m)
      do 203 j=1,n
      do 203 i=1,m
      d(i,j)=(xm-x(i)+dx)*slope
 203  continue
      endif

      if ( itype .eq. 4) then
c*----------------------------------------------------------------------
c*    planar bottom test case
c*----------------------------------------------------------------------
      write(iun(3),*)' input m,n,dx,dy,depth,period'
      read (*,*) m,n,dx,dy,dep,period
      write(iun(3),*) ' input bottom slope'
      read(*,*) xm
      sig=2.*3.1415927/period
      do 251 i=1,m
      do 251 j=1,n
      d(i,j)=dep-xm*float(i-1)*dx
 251  continue
      do 252 i=1,m
 252  x(i)=float(i-1)*dx
      do 253 j=1,n
 253  y(j)=float(j-1)*dy
      endif

      if ( itype .eq. 5) then
c*---------------------------------------------------------------------
c*    radder(1979), configuration 2
c*---------------------------------------------------------------------
      write(iun(3),*)' input m,n,dx,dy,depth'
      read (*,*)m,n,dx,dy,dep
      iu=1
      rad=dep/0.116
      dm=0.1379*dep
      e0=(dep-dm)/rad
      ak0=2.*3.1415927/(0.288*rad)
      sig2=9.806*ak0*tanh(ak0*dep)
      sig=sqrt(sig2)
      do 301 i=1,m
      x(i)=float(i-1)*dx
 301  continue
      do 302 j=1,n
      y(j)=float(j-1)*dx
 302  continue
      do 303 i=1,m
      do 303 j=1,n
      r=sqrt(((x(i)-x(ifix(rad/dx)+1))**2.)+((y(j)-y((n+1)/2)
     1)**2.))
```

```
      if(r.gt.rad)d(i,j)=dep
c*    if(r.le.rad)d(i,j)=dm+e0*r*r
      if(r.le.rad)d(i,j)=dm+e0*r
 303  continue
      endif

      if ( itype .eq. 6) then
c*--------------------------------------------------------------------
c*    grazing incidence on caustic (kirby and dalrymple, 1983)
c*--------------------------------------------------------------------
      write(iun(3),*)' input m,n,dx,dy,depth,period'
      read(*,*)m,n,dx,dy,dep,per
      pi=3.1415927
      sig=2.*pi/per
      d2=2.*dep
      alph=atan(0.02)
      thet=25.*pi/180.
      b=(d2-dep)/tan(alph)
      tt=tan(thet)
      do 351 j=1,n
      y(j)=float(j-1)*dy
 351  continue
      do 352 i=1,m
      x(i)=float(i-1)*dx
 352  continue
      do 353 i=1,m
      do 353 j=1,n
      if(y(j).lt.(y(n)-x(i)*tt))d(i,j)=dep
      if(y(j).ge.(y(n)-x(i)*tt))d(i,j)=dep+cos(thet)*tan(alph)
     1*(x(i)*tt+y(j)-y(n))
      if(y(j).gt.(y(n)-x(i)*tt+b/cos(thet)))d(i,j)=d2
 353  continue
      endif

      if ( itype .eq. 7) then
c*-------------------------------------------------------------------
c*    whalin's channel (1971)
c*-------------------------------------------------------------------
      write(iun(3),401)
 401  format(' whalins channel, input wave period')
      read (*,*)period
      write (iun(3),*)period
      m=100
      n=74
      iu=1
      dx=.242424242
      dy=.33866666/4.
      pi=3.1415927
      sig=2.*pi/period
      do 402 i=1,m
 402  x(i)=float(i-1)*dx
      do 403 j=1,n
 403  y(j)=float(j-1)*dy-dy/2.
      do 404 j=2,n-1
```

```
      g=sqrt(y(j)*(6.096-y(j)))
      do 404 i=1,m
      if(x(i).lt.(10.67-g))d(i,j)=0.4572
      if((x(i).ge.(10.67-g)).and.(x(i).le.(18.29-g)))d(i,j)=
     10.4572+(10.67-g-x(i))/25.
      if(x(i).gt.(18.29-g))d(i,j)=0.1524
 404  continue
      do 405 i=1,m
      d(i,1)=d(i,2)
      d(i,n)=d(i,n-1)
 405  continue
      endif

      if ( itype .eq. 8) then
c*--------------------------------------------------------------------
c*    surface piercing breakwater
c*--------------------------------------------------------------------
      write(*,*)' input m,n,dx,dy'
      read(*,*) m,n,dx,dy
c*    breakwater tip radius
      xt=1.5
      yt=4.
      do 451 j=1,n
      y(j)=float(j-1)*dy
 451  continue
      do 452 i=1,m
      x(i)=float(i-1)*dx
      do 452 j=1,n
      if (x(i).lt.xt) then
        r=sqrt((x(i)-xt)**2+(y(j)-yt)**2)
        dep=.66*r-.37
      else
        dep=.66*abs(y(j)-yt)-.37
      endif
      if(dep.gt..36) dep=.36
      d(i,j)=dep
 452  continue
      endif

      if ( itype .eq. 9) then
c*-------------------------------------------------------------------
c*    generate a pair of breakwaters with rounded heads on each
c*    side of a channel.
c*    xt= beginning location for trunk of breakwater
c*    sl= slope of the sides of the breakwater
c*-------------------------------------------------------------------
      write(*,*) ' input m,n,dx,dy,xt,sl,do'
      read(*,*) m,n,dx,dy,xt,sl,do
      w=float(n-1)*dy
      do 503 i=1,m
         x(i)=float(i-1)*dx
         if (x(i).le.xt) then
            do 501 j=1,n
               y(j)=float(j-1)*dy
```

```
              if (j.lt.n/2) then
                r=sqrt((x(i)-xt)**2+y(j)**2)
                if(r.eq.0.) then
                  cr=0.
                else
                  cr=abs((x(i)-xt))/r
                endif
              dep=sl*r+.05
c*    add bulbous head
c              dep=dep-sl*sqrt(r)*cr
              else
                r=sqrt((x(i)-xt)**2+(y(j)-w)**2)
                if(r.eq.0.) then
                    cr=0.
                else
                    cr=abs((x(i)-xt))/r
                endif
              dep=sl*r+.05
c    add bulbous head
c             dep=dep  -sl*sqrt(r)*cr
              endif
              if (dep.gt.do)dep = do
              if (dep.lt..05)dep = .05
              d(i,j)=dep
 501      continue
        else
          do 502 j=1,n
              if (j.lt.n/2) then
                 yt=0.
              else
                 yt=w
              endif
            dep=sl*abs(y(j)-yt)+.05
            if (dep.gt.do)dep = do
            d(i,j)=dep
 502      continue
        end if
 503  continue
      endif

      if ( itype .eq. 10) then
c*-------------------------------------------------------------------
c*    generate a breakwater with rounded head
c*    with an orientation alpha  (degrees) to the x axis
c*    xh, yh= locus of breakwater head;
c*    sl= slope of the sides of the breakwater;
c*    do= constant depth section.
c*-------------------------------------------------------------------
      write(*,*) ' input m,n,dx,dy,xh, yh,alpha,sl,do'
      read(*,*) m,n,dx,dy,xh,yh,alpha,sl,do
      al=alpha*3.1415927/180.
      write(*,*)' alpha =',al
      co=cos(al)
      si=sin(al)
```

```
      do 551 i=1,m
        x(i)=float(i-1)*dx
           do 551 j=1,n
             y(i)=float(j-1)*dy
             xp=(x(i)-xh)*co+(y(i)-yh)*si
             yp=-(x(i)-xh)*si +(y(i)-yh)*co
c            see if we are in front of the trunk
             if(xp.lt.0.0) then
                r=sqrt(xp*xp+yp*yp)
                dep=sl*r-10.
             else
                dep=sl*abs(yp)-10.
             endif
             if (dep.gt.do) dep=do
             d(i,j)=dep
 551  continue
      endif

      return
      end


c*----------------------------------------------------------------------
c*    error function, hasting's method
c*----------------------------------------------------------------------
      function erfjk(x)
      dimension a(5)
      a(1)=0.254830
      a(2)=-0.284497
      a(3)=1.421414
      a(4)=-1.453152
      a(5)=1.061405
      t=1./(1.+0.327591*x)
      erfjk=1.-exp(-(x**2))*t*(a(1)+t*(a(2)+t*(a(3)+t*(a(4)+t*a(5)))))
      return
      end
```

6.5 *surface.f*

```
c*-----------------------------------------------------------------------------
c*
c*    surface.f
c*
c*    This program converts the file (usually surface.dat) containing
c*    an instantaneous snapshot of the water surface at the computational
c*    grid spacing to a regularly spaced ascii file, suitable for directly
c*    reading into Matlab format.
c*
c*    James T. Kirby
c*    Center for Applied Coastal Research
c*    University of Delaware
c*    Newark, DE 19716
c*    kirby@coastal.udel.edu, (302) 831-2438, FAX (302) 831-1228
c*
c*    Last revision 12/22/94.
c*
c*------------------------------------------------------------------------------

      program surface

      include 'param.h'

      integer i,j,k,m,n,nx,ny,iswap
      integer iret, iout
      real surface(iy,iy)
      real x(iy),y(iy),dx,dy,xold(iy),surfold(iy,iy)
      character*255 fileout
      integer idimsizes(2)

      character*255 fname1,fname2,fname3,fname4,fname5,fname6,
     1              fname7,fname8,fname9,fname10,fname11,fname12,
     1              fname13,fname14,fname15,fnamein

      namelist /ingrid/ mr, nr, iu, ntype, icur, ibc, dxr, dyr, dt,
     1                  ispace, nd, iff, isp, iinput, ioutput
     1         /inmd/ md
     1         /fnames/ fname1,fname2,fname3,fname4,fname5,fname6,
     1                  fname7,fname8,fname9,fname10,fname11,fname12,
     1                  fname13,fname14,fname15
     1         /waves1a/ iwave, nfreqs
     1         /waves1b/ freqs, tide, nwavs, amp, dir
     1         /waves1c/ thet0, freqs, tide, edens, nwavs, nseed
     1         /waves2/ freqin, tidein

      open(8,file='indat.dat')
      read(8,nml=fnames)

      open(10,file=fname6)

c  Enter output file name.
```

```
      write(*,*) 'enter output file name in single quotes'
      read(*,*) fileout

c  Read number of y-direction points from surface.dat

      read(10,*) ny
      read(10,*) (y(j),j=1,ny)

      write(*,*) ' number of y points = ', ny
      write(*,*) ' maximum y = ', y(ny)

c  Read surface data.

      do 10 i=1,100000

      read(10,*) xold(i)

      if (xold(i).lt.0) go to 20

      read(10,*) (surfold(i,j),j=1,ny)

 10   continue

 20   continue

      m=i-1

      write(*,*) ' number of x points in file = ', m
      write(*,*) ' maximum x = ', xold(m)

      dy=y(2)-y(1)
      dx=dy

      write(*,*) ' grid spacing (x and y) in new image = ', dy

      nx=int(xold(m)/dx)+1

      write(*,*) ' number of x points in interpolated image = ', nx

      do 25 j=1,ny
      jj=(ny-j)+1
      surface(1,j)=surfold(1,jj)
 25   continue

      x(1)=0.

      do 40 i=2,nx-1
      x(i)=float(i-1)*dx
      do 35 ii=1,m-1
      if((xold(ii).le.x(i)).and.(xold(ii+1).gt.x(i))) then
          fac=(x(i)-xold(ii))/(xold(ii+1)-xold(ii))
          do 30 j=1,ny
          jj=(ny-j)+1
```

```
        surface(i,j)=(1.-fac)*surfold(ii,jj)+fac*surfold(ii+1,jj)
30      continue
     endif
35   continue
40   continue

     do 45 j=1,ny
     surface(nx,j)=surfold(m,j)
45   continue

     open(11,file=fileout)

     do 50 i=1,nx
     write(11,51)(surface(i,j),j=1,ny)
50   continue

     close(11)

     stop

51   format(500(f10.4))

     end
```

6.6 *refdifplot.m*

```
% refdifplot.m
%
% Script file to read in wave height, wave angle, water depth and surface data
% from refdif1 output,
% and construct various plots.  This
% program uses the quiver routine from Matlab 4.2
%
% James T. Kirby
% Center for Applied Coastal Research
% University of Delaware
% Newark, DE 19716
% kirby@coastal.udel.edu, (302) 831-2438, FAX (302) 831-1228.
%
% 11/27/94

% Read data files.

   load height.dat
   load angle.dat
   load depth.dat

   dx=input(' enter dx:  ');
   dy=input(' enter dy:  ');

   sz=size(height);

% Compute x,y vectors.

   x=dx*(1:sz(1))-dx;
   y=dy*(1:sz(2))-dy;

% Constructing scales arrow plot for wave heights and directions.
% Compute x,y components of arrows.

   DX=height.*cos(pi*angle/180);
   DY=height.*sin(pi*angle/180);

% Now do contours of wave height over depth contours.

   figure(1),clf,hold off
   cf=contour(x,y,height');clabel(cf,'manual'),xlabel('x'),ylabel('y')
   hold on,contour(x,y,depth','--'),axis('equal')

% Now overlay scaled arrows on contours of wave height.

   figure(2),clf,hold off
   cf=contour(x,y,height');clabel(cf,'manual'),xlabel('x'),ylabel('y')
   hold on,quiver(x,y,DX',DY'),axis('equal')

% Now plot the surface.
```

```
isurf=input('do you want to plot the surface? 1=yes:  ');

if isurf ==1, load surf.dat

figure(3),clf,hold off
cf=contour(x,y,surf');clabel(cf,'manual'),xlabel('x'),ylabel('y')
hold on, contour(x,y,depth','--'), axis('equal')

end
```

7 Appendix C: Notes on Using REF/DIF 1 in the LRSS System.

The LRSS user is referred to instructions appearing in the documentation of the program *lrss2rf.f* for creating data files and running *refdif1* in the LRSS system. Additional information is given in the file *README.refdif1* supplied with the program distribution.

7.1 *infile2.f*

```
c*-----------------------------------------------------------------------------
c*  infile2.f
c*
c*  Provide code for reading the namelist file name from the command line
c*  for refdif 1 or refdifs.
c*
c*  igetarg provided on SGI by the link to liblrss, provided by SAIC.
c*
c*  James T. Kirby, October 10, 1994.
c*
c*  LRSS mods made November 22, 1994 by Kurt Schmitt at SAIC
c*
c*  Last revision 11/26/94
c*-----------------------------------------------------------------------------

      subroutine infile(fnamein)
      character*255 fnamein2,fnamein
      integer igetarg
      external igetarg
      character*255 input_namelist
      namelist/lrss_name/fnamein2

c      call lrss_setup

c*   read the filename

      iret=igetarg(1,input_namelist,255)

      if(iret.eq.-1) then
          iret =
     1    error_write('no namelist filename specified on command line')
          call exit(1)
      endif

       open(unit=10,file=input_namelist)
       read(10,lrss_name)
       close(10)
       fnamein=fnamein2
      return
      end
```

7.2 *lrss2rf.f*

```
c*-----------------------------------------------------------------------------
c*    lrss2rf.f
c*
c*
c*    REF/DIF 1 requires two principle input files for data if only the
c*    basic option of running an initially monochromatic, long-crested
c*    wave component is chosen.  The structure of these files is somewhat at
c*    odds with the requirements for input data file structure in LRSS.
c*    The present program is provided to carry out the necessary conversion.
c*
c*    This program takes the single namelist group input file provided by
c*    LRSS, together with the HDF formatted bathymetry file, and generates an
c*    indat.dat file for REF/DIF 1.  This intermediate
c*    step is taken because REF/DIF 1 can generally be initialized in several
c*    different ways, only one of which (running a single monochromatic
c*    component with only wave height, period and direction specified at the
c*    offshore boundary) is provided as an option for LRSS at this point.  The
c*    standard indat.dat file thus has several possible configurations of
c*    namelist groups, only one of which will be utilized here.
c*
c*    It is assumed that:
c*
c*    (1)  The dimensions of the bathymetry grid in the HDF input file
c*         are to be the same as the reference grid dimensions in the
c*         output ascii file refdat.dat and as specified in the output
c*         indat.dat.
c*
c*    (2)  Aside from bathymetry, the user is specifying the wave period,
c*         wave angle, wave height and tidal offset relative to grid datum
c*         for a single long crested monochromatic wave.  Spectral
c*         simulations are handled by the related program REF/DIF S.
c*
c*    (3)  The user tells REF/DIF 1 to generate the data for an image of
c*         the water surface at the computational resolution by setting
c*         the parameter isurface=1.  The user requests data on bottom
c*         velocities at the bathymetry grid resolution by setting the
c*         parameter ibottom=1.
c*
c*    (3)  The program will attempt to determine its own computational
c*         subgridding based on resolution requirements computed in this
c*         program.
c*
c*    (4)  The program will assume that:
c*
c*         - input data is in MKS units.
c*         - the lateral model boundaries will be open and transmitting.
c*         - the composite nonlinear model described in the user's manual
c*           is used.
c*         - frictional dissipation is ignored.
c*         - there will be no user specified subgridding.
c*         - wave-current interaction effects are being neglected.
```

```
c*           (otherwise, the system will need to specify a u,v current
c*           field at the bathymetry grid resolution.)
c*
c*         Deviations from any of these assumptions will require the user
c*         to construct the desired datafileout (=indat.dat) data file
c*         following the instructions in the user's manual.
c*
c*-----------------------------------------------------------------------
c*
c*    Running the program.
c*
c*    The program is to be run with a command line file specifications:
c*
c*    lrss2rf datafilein
c*
c*    Runtime messages are stored in lrss2rf.log.  REF/DIF 1 is then
c*    started using the command line:
c*
c*    refdif1 filein
c*
c*    where 'filein' is a namelist file containing the file name 'datafileout'
c*    in the namelist group 'lrss_name'. (See the code fragment 'infile2.f').
c*
c*    'gridfilein' is the name of the HDF bathymetry file.
c*
c*    'datafilein' is the name of the single namelist group input data file.
c*    A sample of this file follows.
c*
c*-----------------------------------------------------------------------
c*
c*    Sample data file:
c*
c* $input
c* height=1.
c* period=10.
c* angle=0.
c* tideoffset=0.
c* gridfilein='camppend.hdf'
c* datafileout='indat.dat'
c* logfile='lrss2rf.log'
c* fname1='refdat.dat'
c* fname2='outdat.dat'
c* fname3='subdat.dat'
c* fname6='surface.dat'
c* fname7='bottomu.dat'
c* fname8='angle.dat'
c* fname9=' '
c* fname10='refdif1.log'
c* fname11='height.dat'
c* fname12=' '
c* fname13=' '
c* fname14=' '
c* fname15='depth.dat'
c* $end
```

```
c*
c*    The standard names for REF/DIF 1 input files would correspond to:
c*
c*    datafileout = indat.dat
c*    fname1= refdat.dat
c*
c*    but any arbitrary file name may be specified.  Runtime messages are
c*    stored in lrss2rf.log.
c*
c*
c*-----------------------------------------------------------------------
c*
c*
c*    James T. Kirby
c*    Center for Applied Coastal Research
c*    University of Delaware
c*    Newark, DE 19716
c*
c*    (302) 831-2438, FAX (302) 831-1228, kirby@coastal.udel.edu
c*
c*    Last revision 12/21/94.
c*
c*-----------------------------------------------------------------------

      program lrss2rf

      include 'param.h'

      common/indat/ height,period,angle,tideoffset, mr, nr, dxr, dyr
      common/refdat/ bath,nd,dt
      common/fname_s/fname1,fname2,fname3,fname4,fname5,fname6,fname7,
     1              fname8,fname9,fname10,fname11,fname12,fname13,
     1              fname14,fname15

c  Information about hdf file.

      integer i,j,k,m,n,iswap,n3,n4
      integer iret, iwrite, iout, error_write, error_exit
      external itegarg
      real bathflip(ixr,iyr),bath(ixr,iyr),hdfarray(ixr,iyr)
      real x(ixr),y(iyr),dx,dy
      character*255 infile, fileout
      character*255 coordsys
      character*255 labelx,labely,labelf,unitx,unity,unitf
      character*255 formatx,formaty,formatf,filein
      character*255 datafilein, datafileout, gridfilein, logfile
      character*255 fname1,fname2,fname3,fname4,fname5,fname6,fname7,
     1              fname8,fname9,fname10,fname11,fname12,fname13,
     1              fname14,fname15
      integer idimsizes(2)

c*-----------------------------------------------------------------------
c*   First, set up the ability to read the input file and output file
c*   names from the command line.
```

```
c*------------------------------------------------------------------------------

      integer igetarg
      external igetarg

c*-------------------------------------------------------------------------
c*    Namelist structure for datafilein
c*-------------------------------------------------------------------------

      namelist/input/height,period,angle,tideoffset,isurface,ibottom,
     1               gridfilein,datafileout,logfile,
     1              fname1,fname2,fname3,fname6,fname7,
     1              fname8,fname9,fname10,fname11,fname12,fname13,
     1              fname14,fname15

c*-------------------------------------------------------------------------
c*   add this call to work around SGI bug (not needed if program being
c*   used as a free-standing program).
c*-------------------------------------------------------------------------

c      call lrss_setup

c*-------------------------------------------------------------------------
c*   read the filename, check that we got it.
c*-------------------------------------------------------------------------

      iret=igetarg(1,datafilein, 255)

      if(iret1.eq.-1) then
        iwrite = error_write
     1       ('no filename for datafilein specified on command line')
        call exit(1)
      endif

c*------------------------------------------------------------------------
c*    Enter the namelist data from datafilein
c*------------------------------------------------------------------------

      open(5,file=datafilein)
      read(5,nml=input)
      close(5)

      open(10,file=logfile)
c*-----------------------------------------------------------------------
c*   First, we will read in information about the grid from the LRSS
c*   HDF formatted file (specified by gridfilein) and generate the ASCII
c*   file refdat.dat usually used by REF/DIF 1.  It is assumed that the
c*   REF/DIF 1 parameters ixr and iyr are set large enough (in param.h)
c*   to contain the bathymetry grid being read in.
c*-----------------------------------------------------------------------

      write(10,*) ' begin r_hdf'

      call r_hdf(gridfilein,bath,ixr,iyr,mr,nr,x,y,hdfarray,
```

```
     $      labelx,formatx,unitx,labely,formaty,unity,
     $      labelf,formatf,unitf,coordsys,iswap)

      write(10,*) ' end of r_hdf'

c*-----------------------------------------------------------------------
c*    Dimensions mr and nr of actual grid
c*-----------------------------------------------------------------------

      write(10,*) 'mr=',mr
      write(10,*) 'nr=',nr

      dxr=abs(x(2)-x(1))
      dyr=abs(y(1)-y(2))

c*-----------------------------------------------------------------------
c*    Generate the (fname1=refdat.dat) data file.
c*-----------------------------------------------------------------------

      open(8,file=fname1)
      do 10 i=1,mr
      write(8,20) (bath(i,j),j=nr,1,-1)
 10   continue
      close(8)
 20   format(20f10.4)

c*-----------------------------------------------------------------------
c*    Now we must inspect the depth grid to get an estimate of what the
c*    level of automatic grid subdivision should be.  This is done as in
c*    REF/DIF 1 itself.
c*-----------------------------------------------------------------------

      call con

c*-----------------------------------------------------------------------
c*    Now we need to generate the namelist file (usually indat.dat) for
c*    REF/DIF 1.
c*-----------------------------------------------------------------------

      call indatgen(datafileout)

c*-----------------------------------------------------------------------
c*    All done.
c*-----------------------------------------------------------------------

      call exit(0)
      end

c*-----------------------------------------------------------------------
c*    indatgen
c*
c*    Generate the ascii namelist file indat.dat for input into REF/DIF 1
c*
c*-----------------------------------------------------------------------
```

```
      subroutine indatgen(datafileout)

      include 'param.h'

c*---------------------------------------------------------------------
c*    Information from main program.
c*---------------------------------------------------------------------

      common/indat/height,period,angle,tideoffset,mr,nr,dxr,dyr
      common/refdat/bath,nd,dt
      common/fname_s/fname1,fname2,fname3,fname4,fname5,fname6,fname7,
     1              fname8,fname9,fname10,fname11,fname12,fname13,
     1              fname14,fname15

      character*255 datafileout

c*---------------------------------------------------------------------
c*    information for datafileout ( =indat.dat).
c*---------------------------------------------------------------------

      dimension iff(3)
      dimension bath(ixr,iyr)
      dimension freqs(ncomp), nwavs(ncomp)
      dimension amp(ncomp,ncomp), dir(ncomp,ncomp), tide(ncomp)

      character*255 fname1,fname2,fname3,fname4,fname5,fname6,fname7,
     1              fname8,fname9,fname10,fname11,fname12,fname13,
     1              fname14,fname15

c*--------------------------------------------------------------------
c*   The following files correspond to a portion of REF/DIF 1 which is
c*   never used by LRSS, but are required for input into REF/DIF 1.
c*   Names are specified here in order to avoid having the system or user
c*   invent an arbitrary name for a non-existant file.
c*---------------------------------------------------------------------

      data fname4/'fname4.dat'/,fname5/'fname5.dat'/


      namelist /ingrid/ mr, nr, iu, ntype, icur, ibc, dxr, dyr, dt,
     1                  ispace, nd, iff, isp, iinput, ioutput
     1         /fnames/ fname1,fname2,fname3,fname4,fname5,fname6,
     1                  fname7,fname8,fname9,fname10,fname11,fname12,
     1                  fname13,fname14,fname15
     1         /waves1a/ iwave, nfreqs
     1         /waves1b/ freqs, tide, nwavs, amp, dir

c*---------------------------------------------------------------------
c*    Define any variables not yet specified
c*---------------------------------------------------------------------

      iu=1
      ntype=1
```

```
      icur=0
      ibc=1
      ispace=0
      iff(1)=0
      iff(2)=0
      iff(3)=0
      isp=0
      iinput=1
      ioutput=1

      iwave=1
      nfreqs=1
      freqs(1)=period
      tide(1)=tideoffset
      nwavs(1)=1
      amp(1,1)=height/2.
      dir(1,1)=angle


c*-----------------------------------------------------------------------
c*    Open the data file and write the namelists.
c*-----------------------------------------------------------------------

      open(7,file=datafileout)

      write(7,nml=fnames)
      write(7,nml=ingrid)
      write(7,nml=waves1a)
      write(7,nml=waves1b)

      close(7)

c*-----------------------------------------------------------------------
c*    datafileout (=indat.dat) all done.
c*-----------------------------------------------------------------------

      return
      end

c*-----------------------------------------------------------------------
c*    con
c*
c*    Compute wavenumbers and number of waves per grid space in order to
c*    estimate the required level of subdivision.
c*
c*-----------------------------------------------------------------------

      subroutine con

      include 'param.h'

      common/indat/ height,period,angle,tideoffset, mr, nr, dxr, dyr
      common/refdat/ bath,nd,dt
```

```
      dimension bath(ixr,iyr),md(ixr)
      real k(ixr,iyr),kb(ixr)

      mdmax=1
      pi=3.1415927
      eps=1.0e-05
      omega=2.*pi/period

c*-----------------------------------------------------------------------
c*    Compute x-direction subdivisions just as in REF/DIF 1
c*-----------------------------------------------------------------------

      depthmin=0.
      depthmax=0.

      do 100 i=1,mr-1

      npts=0
      sumk=0.

      do 13 j=1,nr
      dref=bath(i,j)+tideoffset
      if(dref.gt.depthmax)depthmax=dref
      if(dref.lt.0.001) dref=0.001
      call wvnum(dref,0.0,omega,k(i,j),eps,icdw,i,j)
      if(dref.gt.0.05) then
        sumk=sumk+k(i,j)
        npts=npts+1
      endif
 13   continue
      if(npts.eq.0)then
      kb(i)=k(i,1)
      else
      kb(i)=sumk/float(npts)
      endif
      alw=2.*pi/kb(i)
      anw=dxr/alw
      np=ifix(5.*anw)
      if(np.lt.1) np=1

      if(npts.gt.0) then
      md(i)=min((ix-1),np)
      if(np.gt.(ix-1)) write(10,200) i
      else
      md(i)=md(i-1)
      endif

 100  continue

      dt=depthmax

c*----------------------------------------------------------------------
c*    Find the biggest one, compute the corresponding dx and dy, and
c*    compute the resulting nd.
```

```
c*-----------------------------------------------------------------------

      do 101 i=1,mr-1
      if(md(i).gt.mdmax) mdmax=md(i)
 101  continue

      write(10,*) ' md(ir) max = ', mdmax

      dx=dxr/float(mdmax)

      write(10,*) ' dxmin = ', dx
      dy=dx
      nd=ifix(dyr/dy)

      return

 200  format(' model tried to put more spaces than allowed in',
     1' grid block ',i3)

      end

c*-----------------------------------------------------------------------
c*   wvnum
c*
c*   Compute wavenumbers.
c*
c*-----------------------------------------------------------------------

      subroutine wvnum(d,u,s,k,eps,icdw,i,j)

      include 'param.h'

      real k,kn

c  Constants.

      g=9.806
      pi=3.1415927
      k=s*s/(g*sqrt(tanh(s*s*d/g)))

c  Newton-Raphson iteration.

      do 1 ii=1,20
      f=s*s-2.*s*k*u+k*k*u*u-g*k*tanh(k*d)
      fp=-2.*s*u+2.*k*u*u-g*tanh(k*d)-g*k*d/(cosh(k*d)**2.)
      kn=k-f/fp
      if((abs(kn-k)/kn).lt.eps)go to 2
      k=kn
 1    continue
      t=2.*pi/(sqrt(g*k*tanh(k*d))+k*u)
      write(10,100)i,j,k,u,d,f,t
      icdw=1

      return
```

```
2     k=kn

      return

100   format(' wavenumber iter. failed to converge on row',i10,
     1'  column',i10/
     1'      k=',f15.8,'      u=',f15.8/
     1'      d=',f15.8,'      f=',f15.8/
     1'      t=',f15.8)

      end
```

7.3 *rf2hdf.f*

```
c*-----------------------------------------------------------------------------
c*
c*    rf2hdf.f
c*
c*    This program converts selected refdif1 files at the reference
c*    grid spacing to hdf format, for use by other LRSS algorithms and
c*    by the Matlab Image Processing Toolbox.
c*
c*    James T. Kirby
c*    Center for Applied Coastal Research
c*    University of Delaware
c*    Newark, DE 19716
c*
c*    kirby@coastal.udel.edu, (302) 831-2438, FAX (302) 831-1228.
c*
c*    Last revision 12/22/94.
c*
c*------------------------------------------------------------------------------

      program rf2hdf

      include 'param.h'

      dimension iff(3)

      common/surf/surface(iy,iy)

      integer nx,ny, iwrite
      integer iret, igetarg, error_write
      external itegarg
      real out_array(ixr,iyr),hdfarray(ixr,iyr)
      real x(ixr),y(iyr),dxr,dyr

      character*255 fname1,fname2,fname3,fname4,fname5,fname6,
     1              fname7,fname8,fname9,fname10,fname11,fname12,
     1              fname13,fname14,fname15,fnamein
      character*255 coordsys
      integer iswap

      namelist /ingrid/ mr, nr, iu, ntype, icur, ibc, dxr, dyr, dt,
     1                  ispace, nd, iff, isp, iinput, ioutput
     1         /inmd/ md
     1         /fnames/ fname1,fname2,fname3,fname4,fname5,fname6,
     1                  fname7,fname8,fname9,fname10,fname11,fname12,
     1                  fname13,fname14,fname15
     1         /waves1a/ iwave, nfreqs
     1         /waves1b/ freqs, tide, nwavs, amp, dir
     1         /waves1c/ thet0, freqs, tide, edens, nwavs, nseed
     1         /waves2/ freqin, tidein

      open(6,file='rf2hdf.log')
```

```
      call infile(fnamein)

      open(8,file=fnamein)

      read(8,nml=fnames)
      read(8,nml=ingrid)

c  Compute grid spacing data.

      do 1 i=1,mr
      x(i)=float(i-1)*dxr
 1    continue

      do 2 j=1,nr
      y(j)=float(j-1)*dyr
 2    continue

c*-----------------------------------------------------------------------
c    values used for labels, formats, units are your choice
c    coordsys is usually just null string = ''
c    iswap is usually 0
c*-----------------------------------------------------------------------
      coordsys=' '
      iswap=0

c*-----------------------------------------------------------------------
c   if output file names are not null strings, then
c   write out data in hdf format
c*-----------------------------------------------------------------------
c   angle file output
c*-----------------------------------------------------------------------

      if (fname8.ne.' ') then
        write(6,*) 'in angleout w_hdf'
write(6,*) 'iyr,mr,nr=',iyr,mr,nr
        open(10,file=fname8)
        do 10 i=1,mr
        read(10,*) (out_array(i,j),j=nr,1,-1)
 10     continue
        close(10)
        print*, 'angle=',out_array(1,1),out_array(100,100),
     1          out_array(mr,nr)


        call w_hdf(fname8,out_array,ixr,iyr,mr,nr,x,y,hdfarray,
     $    'x axis','f8.1','meters','y axis','f8.1','meters',
     $    'reference_grid_angles','e10.3','degrees',coordsys,iswap)

      endif

c*-----------------------------------------------------------------------
c   height file output
```

```
c*-----------------------------------------------------------------------

      if (fname11.ne.' ') then

        open(10,file=fname11)

        do 20 i=1,mr
        read(10,*) (out_array(i,j),j=nr,1,-1)
 20     continue
        close(10)
        write(6,*) 'in height w_hdf'
write(6,*) 'iyr,mr,nr=',iyr,mr,nr
        write(6,*) 'height=',out_array(1,1),out_array(100,100),
     1           out_array(mr,nr)

        call w_hdf(fname11,out_array,ixr,iyr,mr,nr,x,y,hdfarray,
     $    'x axis','f8.1','meters','y axis','f8.1','meters',
     $    'wave height','e10.3','meters',coordsys,iswap)

      endif

c*-----------------------------------------------------------------------
c  Read radiation stress data.
c*-----------------------------------------------------------------------

      if (fname12.ne.' '.and.fname13.ne.' '.and.fname14.ne.' ') then
        write(6,*) 'in stress out'
        open(10,file=fname12)

        do 30 i=1,mr
        read(10,*) (out_array(i,j),j=nr,1,-1)
 30     continue
        close(10)

        call w_hdf(fname12,out_array,ixr,iyr,mr,nr,x,y,hdfarray,
     $    'x axis','f8.1','meters','y axis','f8.1','meters',
     $    'radiation stress sxx','e10.3','kg/sec^2',coordsys,iswap)

        open(10,file=fname13)
        do 40 i=1,mr
        read(10,*) (out_array(i,j),j=nr,1,-1)
 40     continue
        close(10)

        call w_hdf(fname13,out_array,ixr,iyr,mr,nr,x,y,hdfarray,
     $    'x axis','f8.1','meters','y axis','f8.1','meters',
     $    'radiation stress sxy','e10.3','kg/sec^2',coordsys,iswap)

        open(10,file=fname14)
        do 50 i=1,mr
        read(10,*) (out_array(i,j),j=nr,1,-1)
 50     continue
        close(10)
```

```
        call w_hdf(fname14,out_array,ixr,iyr,mr,nr,x,y,hdfarray,
     $    'x axis','f8.1','meters','y axis','f8.1','meters',
     $    'radiation stress syy','e10.3','kg/sec^2',coordsys,iswap)
       endif

c*----------------------------------------------------------------------------
c*   surface data.
c*----------------------------------------------------------------------------

      if (fname6.ne.' ') then
write(6,*) 'in surface out'
call surface2hdf(fname6,nx,ny,logfileout)
        call w_hdf(fname6,surface,ixr,iyr,nx,ny,x,y,hdfarray,
     $    'x axis','f8.1','meters','y axis','f8.1','meters',
     $    'surface_image','e10.3','kg/sec^2',coordsys,iswap)
      endif
      call exit(0)
      end

c*---------------------------------------------------------------------------
c*    surface
c*
c*   Interpolate the surface image onto a regular grid.
c*---------------------------------------------------------------------------

      subroutine surface2hdf(fname6,nx,ny,logfileout)

      include 'param.h'

      common/surf/surface(iy,iy)

      integer i,j,m,nx,ny
      real x(iy),y(iy),dx,dy,xold(iy),surfold(iy,iy)

      character*255 fname6, logfileout

      open(10,file=fname6)

c  Enter output file name.

c  Read number of y-direction points from surface.dat

      read(10,*) ny
      read(10,*) (y(j),j=1,ny)

      write(6,*) ' number of y points = ', ny
      write(6,*) ' maximum y = ', y(ny)

c  Read surface data.

      do 10 i=1,100000

      read(10,*) xold(i)
```

```
      if (xold(i).lt.0) go to 20

      read(10,*) (surfold(i,j),j=1,ny)

10    continue

20    continue
      close(10)
      m=i-1

      write(6,*) ' number of x points in file = ', m
      write(6,*) ' maximum x = ', xold(m)

      dy=y(2)-y(1)
      dx=dy

      write(6,*) ' grid spacing (x and y) in new image = ', dy

      nx=int(xold(m)/dx)+1

      write(6,*) ' number of x points in interpolated image = ', nx

      do 25 j=1,ny
      jj=(ny-j)+1
      surface(1,j)=surfold(1,jj)
25    continue

      x(1)=0.

      do 40 i=2,nx-1
      x(i)=float(i-1)*dx
      do 35 ii=1,m-1
      if((xold(ii).le.x(i)).and.(xold(ii+1).gt.x(i))) then
          fac=(x(i)-xold(ii))/(xold(ii+1)-xold(ii))
          do 30 j=1,ny
          jj=(ny-j)+1
          surface(i,j)=(1.-fac)*surfold(ii,jj)+fac*surfold(ii+1,jj)
30        continue
      endif
35    continue
40    continue

      do 45 j=1,ny
      jj=(ny-j)+1
      surface(nx,j)=surfold(m,jj)
45    continue

       close(11)
       return
       end
```

8 Appendix D: Frequently Asked Questions

1. What is the maximum current this model can handle?

   The program can handle currents that are of the same order as the wave speed. However, opposing currents which are strong enough to stop the waves cause a singularity in the wave height due to the absence of reflection effects and no provision for steepness-limited breaking. These extensions will be added in the future.

2. In which situations should the user specify totally reflecting lateral boundaries ($ibc = 0$), and when should partially transmitting lateral boundaries ($ibc = 1$) be used?

   It is common practice to use the closed boundary condition as the amount of reflection can be identified and the domain width can be chosen so that there are no reflections in the region of interest. The final run may be made using partially transmitting boundaries, but it should be noted that there will still be a certain amount of reflected waves present in the domain.

3. What is the difference between user-specified subdivisions and user-specified subgrid?

   User-specified subdivisions and user-specified subgrids are two completely independent options. The user has to specify the desired number of subdivisions in the y-direction nd. However, the user has an option between specifying the number of subdivisions in the x-direction $md(ir)$ or having the program specify those. If the program specifies $md(ir)$, the switch *ispace* has to be set to zero and the program performs interpolations of the depth and velocity grids. If the user choses to specify the $md(ir)$ values himself, the switch *ispace* has to be set to unity. In this case, the user again has a choice: The user can specify $isp = 0$, then the program will perform interpolations. The user can set $isp = 1$, in which case he/she has to specify a subgrid in the file *subdat.dat*.

4. What are the guidelines for the specification of the number of subdivisions $md(ir)$ in the x-direction?

   It is recommended that the number of subdivisions $md(ir)$ be determined by the program. However, if the user wants to specify $md(ir)$'s, the user should make sure that the subdivided grid is at least as fine as the program would have determined. In order to achieve this, the user can first run the program by letting it pick its own subdivisions. Then the user can chose his/her subdivision using the programs subdivisions as a guideline.

5. What are the guidelines for the choice of the number of subdivisions nd in the y-direction ?

   nd should be chosen such that the final subdivided reference grid cells have increments in the x- and y-directions that are fairly close in size. This is best determined by running REF/DIF 1 once to check how many x-direction subdivisions are introduced in the most finely subdivided grid, and then subdividing in y by a corresponding amount

6. Where is the origin of the grid ?

   The origin of the domain is always chosen to be the lower right-hand corner of the domain with the x-axis pointing in the propagation direction and the y-axis pointing towards the left.